NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

THREE-DIMENSIONAL PERSPECTIVE IMAGE
GENERATION FROM SONAR BATHYMETRY
AND IMAGERY DATA

by

Robert J. Myers

June 1988

Thesis Advisor: Chin-Hwa Lee

Approved for public release; distribution is unlimited

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release; distribution is unlimited
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5 MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Naval Postgraduate School 6b OFFICE SYMBOL (*If applicable*) 62	7a. NAME OF MONITORING ORGANIZATION Naval Postgraduate School
6c. ADDRESS (*City, State, and ZIP Code*) Monterey, California 93943-5000	7b. ADDRESS (*City, State, and ZIP Code*) Monterey, California 93943-5000
8a. NAME OF FUNDING / SPONSORING ORGANIZATION 8b OFFICE SYMBOL (*If applicable*)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (*City, State, and ZIP Code*)	10 SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO / TASK NO / WORK UNIT ACCESSION NO.

11. TITLE (*Include Security Classification*) THREE-DIMENSIONAL PERSPECTIVE IMAGE GENERATION FROM SONAR BATHYMETRY AND IMAGERY DATA

12. PERSONAL AUTHOR(S)
Robert J. Myers

13a. TYPE OF REPORT	13b TIME COVERED	14. DATE OF REPORT (*Year, Month, Day*)	15 PAGE COUNT
Master's Thesis	FROM _____ TO _____	1988 June	109

16. SUPPLEMENTARY NOTATION The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

17. COSATI CODES			18 SUBJECT TERMS (*Continue on reverse if necessary and identify by block number*)
FIELD	GROUP	SUB-GROUP	3-D Image Generation, Sonar Images, Perspective Views

19. ABSTRACT (*Continue on reverse if necessary and identify by block number*) This thesis develops a program which will merge or overlay imagery and terrain elevation data and create a synthetic 3-D perspective view of the ocean bottom. The observer may position himself at various locations and see the terrain from different viewpoints. The elevation data is grouped into triangular panels and the color information is averaged from the imagery data file. The entire panel is assigned a single color equal to the average. These panels are then projected onto an image plane by using a 3D to 2D perspective transformation. Hidden surfaces are removed by a "painters" algorithm which relies on sorting the panels based on distance from the observer.

20 DISTRIBUTION / AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL Chin-Hwa Lee	22b. TELEPHONE (*Include Area Code*) (408) 646-2056 22c. OFFICE SYMBOL 62Le

DD FORM 1473, 84 MAR 83 APR edition may be used until exhausted
All other editions are obsolete

☆ U.S. Government Printing Office: 1986—606-243

Approved for public release; distribution is unlimited.

Three-Dimensional Perspective Image Generation from Sonar Bathymetry and Imagery Data

by

Robert J. Myers
Lieutenant, United States Navy
B.S., United States Naval Academy, 1978

Submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
June 1988

ABSTRACT

This thesis develops a program which will merge or overlay imagery and terrain elevation data and create a synthetic 3-D perspective view of the ocean bottom. The observer may position himself at various locations and see the terrain from different viewpoints. The elevation data is grouped into triangular panels and the color information is averaged from the imagery data file. The entire panel is assigned a single color equal to the average. These panels are then projected onto an image plane by using a 3D to 2D perspective transformation. Hidden surfaces are removed by a "painters" algorithm which relies on sorting the panels based on distance from the observer.

TABLE OF CONTENTS

I. INTRODUCTION

The goal of this thesis is to develop the software required to display a 3-D perspective view of combined sonar imagery and bathymetric data. The result is a synthesized image representing a 3-D perspective view of the terrain from a given observer location.

The issues involved include: first, how can the sonar imagery data and the bathymetric data be combined; second, what is the methodology of creating the 3-D perspective view from all observer locations; third, how fast can each view be generated using real data and fourth, how is the picture quality affected by the resolution of the data.

Conventional methods of displaying elevation data are with contour lines or 3-D grid line drawings. Some applications merge the contour lines with color data and this aids in the overall comprehension of the data. Recently the speed of the digital computer has been called upon to generate 3-D views based on elevation with shading. The shading value may be based on the elevation, or may be based on other information available. These displays are an improvement over the simple contour plots, as well as the 3-D grid line drawings. The advantage of

any 3-D perspective display method is the ability to "see" the terrain in a fashion which appears normal to the observer.

Applications of this method range from low cost flight simulators for manned or unmanned vehicles to real time displays of multiple source information in a fashion which results in a greater level of comprehension or interpretation. For example, a combat aircraft pilot must be able to assimilate vast amounts of data. The pilot must search multiple electronic displays in order to put together a picture of the environment in his mind. The pilot must answer several questions simultaneously: Where am I?, Where are my friends?, Where is the enemy?, What is the condition of my aircraft? Simplifying the presentation of this information in a simulated perspective view of the surrounding environment will improve the response of the pilot.[Ref. 1, p. 64] The ability to combine the various data types into a combined display is a form of "sensor fusion". In the aircraft example the data may be video from external video cameras and radar or infrared sensor returns.

Generally, sensor fusion refers to the ability to merge layers of information from various sources into a new form. The data may be imagery data from Landsat or aircraft, or any other layer of data within the same geographical area such as magnetic anomaly or infrared

response data. These various types of data must be in a form which can be represented by a specific color or gray shade. Adding color or shading from the other data source allows the correlation between it and the terrain elevation data to become apparent. A specific area of interest includes shipboard systems where numerous sensors are gathering data and the operator selectively views the output from a single sensor or a combination of sensors. If the outputs from multiple sensors are combined into a 3-D perspective view of the surrounding environment, the response time of the operator will improve. By utilizing the 3-D techniques, a view forward of the ship could be displayed on a CRT screen. Combining radar elevation data with an image file database would permit the formation of a 3-D perspective view of the radar image. This same technique used with the sonar data obtained by the ship's sonar systems and imagery data files of the harbor channel would allow a realistic view of the channel and permit more efficient usage of the available sonar systems.

The software developed in this thesis overlay side-looking sonar imagery data of the ocean bottom onto the bathymetric data of the same geographic area. This is an extension of work by L. Coleman [Ref. 2]. Coleman's work concentrated in generating a 3-D perspective view of land terrain. Several items added in this work include the ability to approach the area from any heading (observer

locations within the area boundaries are permitted), reduction of the time to create one image, and drawing of the image is included in the main routine. In addition to the work by Coleman, a separate project by McGhee, Zyda, Smith and Streyle incorporates many of the same techniques [Ref. 3]. The imagery data is in the form of a 512 x 512 pixel image with 256 possible gray levels. The bathymetric data is gridded and consists of depth values on a latitude longitude grid. The bathymetric data, called the elevation data, is segmented into triangular panels. The boundaries of each panel overlay the image data, image pixels inside the panel are averaged and this average gray level is assigned to the entire panel. To generate the perspective view, each elevation point is projected onto an image plane located at the observer location, and oriented with respect to the observer course. This projection is accomplished through a 3D-2D perspective projection transformation which maps the elevation point coordinates to image plane coordinates. The synthesized image is generated on a monitor and approximates the view from the observer position with a 38.6 degree field of view. The synthesized view is directly affected by the resolution of the input data. The elevation file resolution affects both the elevation drawing and the shading of the image. Shading is affected due to the averaging of pixel shades within each triangle. The drawing is affected by the low

sampling rate of the terrain; only the larger features will be present in the displayed image.

The software is implemented in FORTRAN on a DEC Micro-Vax GPX II. This system is capable of 1024 x 864 pixel resolution, with 256 colors or 256 gray levels. A graphics software package from DEC, MicroVMS Workstation Graphics Software version 3.0, is used to draw and fill each panel on the monitor. The software may be run on other systems, with changes required in the screen plotting routines and the number of shades or color selection. Also the size of the input data files may be limited on other systems which do not have sufficient memory to store the input data in arrays. Results of this work show that the performance is limited by the plotting speed of the Micro-Vax. The calculation of the perspective image is approximately 11% of the total time required to complete the process. The remainder of time is used to draw the image on the screen.

Using this software, an observer may select a position and heading and "see" the ocean bottom terrain in a 3-D perspective form. The observer heading is not limited, the position may be within the boundaries of the area selected, and the view is generated directly on the screen. Several options are available including a magnification factor, elevation scale exaggeration, saving the image to disk and the ability to respecify a new observer location. The speed of image generation is

directly affected by the resolution or size of the input data files, using 61 x 61 elevation points, an image is generated in 40 seconds.

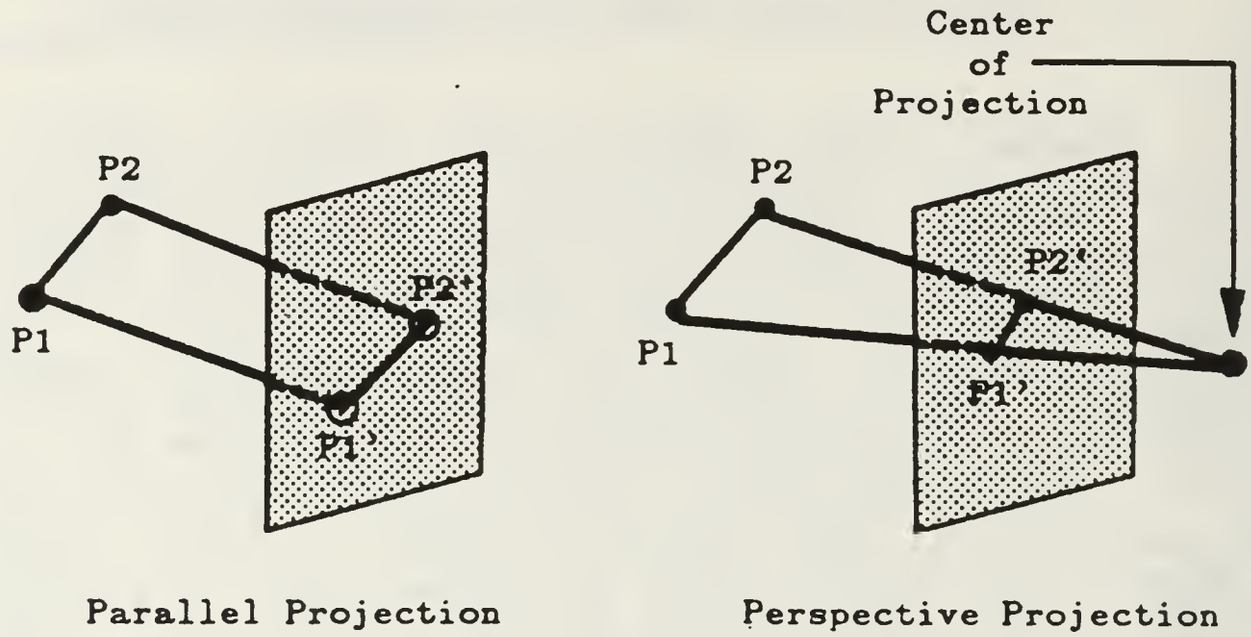
Chapter Two will cover the basic image geometry in order to develop the perspective projection transformation equation, and the orientation of the image plane. Chapter Three details the data file formats, resolution and how they are prepared for use in the main routine. Chapter Four covers the implementation of the algorithms in the program and any alternatives which were investigated. Chapter Five details program operation and Chapter Six presents conclusions and several recommendations for improvement or further study.

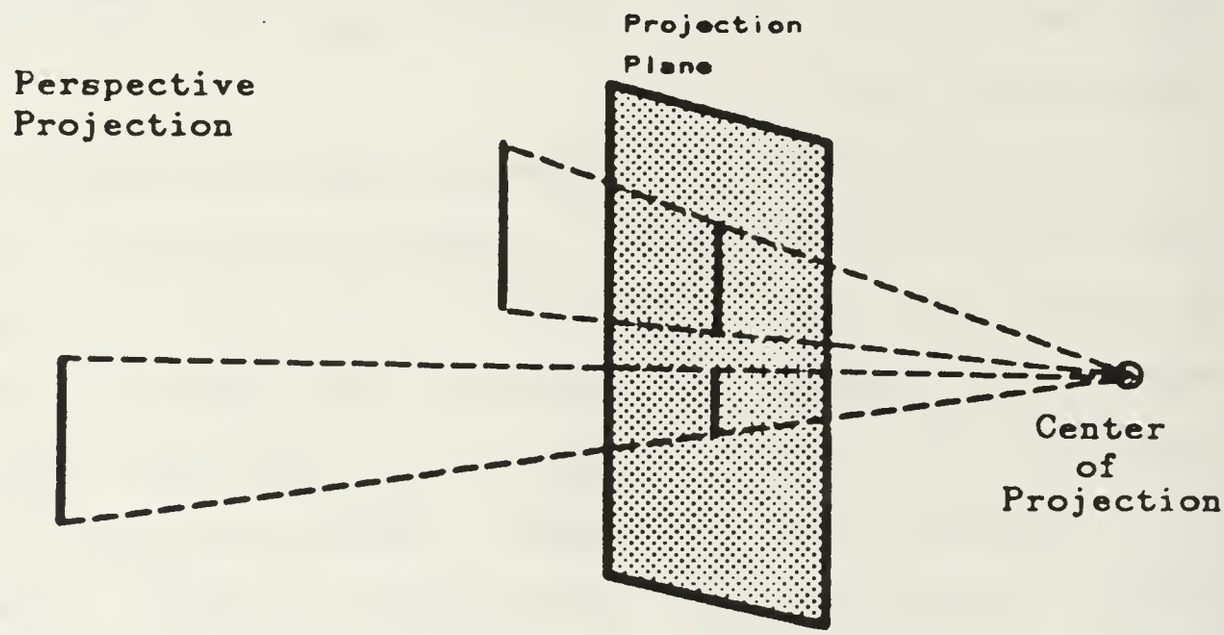
II. IMAGE FORMATION USING TRANSFORMATION GEOMETRY

Inherent in this project is the ability to take the two dimensional gridded elevation data which represents a three dimensional object and display a perspective view on a two dimensional monitor. Two basic approaches, namely, parallel or perspective projection can accomplish this.

A parallel projection is one where the points of the object are projected onto the image plane by parallel lines, that is the projection lines do not converge. This method has the advantage of maintaining relative dimensions of the object and is useful for obtaining measurements [Ref. 3, p. 52].

In the perspective projection all points are projected onto the image plane through a reference point which will be called the focal point [Ref. 4, p. 133]. In this method, the relative dimensions of the object are not preserved, but the image appears more realistic. Figure 2-1 illustrates the two methods. Because the goal is feature recognition and interpretation through screen display and not precise measurement, the perspective projection is used.





Closer lines appear larger than more distant lines of equal length.

Figure 2-1 Parallel and Perspective Projections [Ref. 3, p. 54]

The perspective projection may be generalized as a reference 3-D to 2-D transformation and is illustrated in matrix notation:

$$\bar{A} = [\, M \,]\, \bar{B} \qquad (2.1)$$

where: $\bar{A}$-represents a vector in the image 3D coordinate system

$\bar{B}$-represents a vector in the object 3D coordinate system

$\bar{M}$-represents the transformation matrix

A. COORDINATE SYSTEMS

Equation 2.1 refers to the image 3-D coordinate system and the object 3-D coordinate system. The object is located in a right-handed 3-D cartesian coordinate system where each object point is described in terms of (X,Y,Z) geo-rectangular coordinates. The center of the earth is located at (0,0,0), the X-axis points to the intersection of 0 degrees latitude and 0 degrees longitude, the Y-axis points to the intersection of 0 degrees latitude and 90 degrees east longitude and the Z-axis points to true north. Figure 2-2 illustrates this coordinate system. There is a direct relationship between the latitude, longitude, height with respect to sea level and the (X,Y,Z) object coordinates. The input data is supplied on a latitude, longitude grid and is converted to (X,Y,Z)

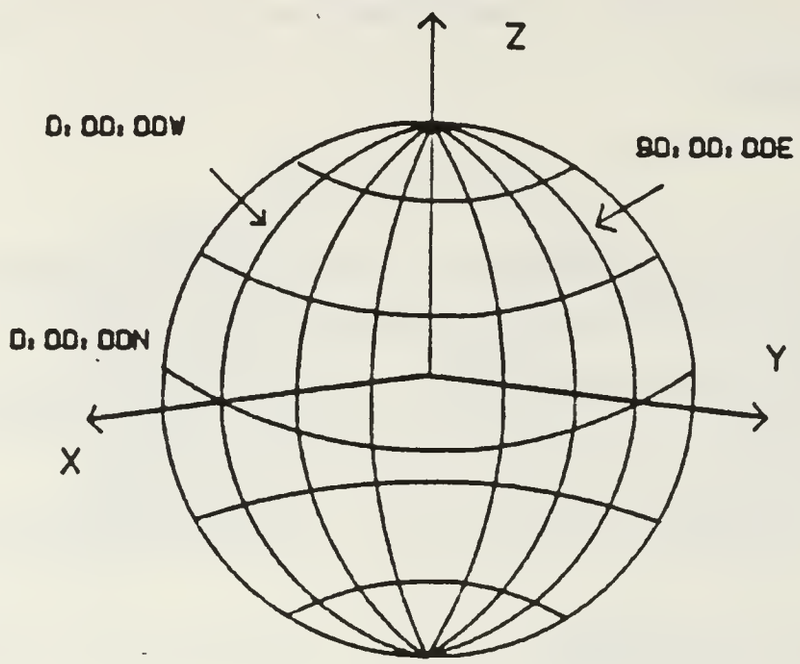


Figure 2-2 Object Coordinate System

coordinates for use in the transformation equations. The image coordinate system is also a right-handed 3-D cartesian system. The focal point is located at (x0,y0,F). Figure 2-3 illustrates the image coordinate system.

B. 3-D PERSPECTIVE TRANSFORMATION

The elements of the [M] matrix in equation 2.1 are the cosines of the spatial angles that each of the image system axis x,y,z make with each of the object system axis X,Y,Z. This substitution results in the following specific form for the matrix [M]. [Ref. 4, p. 139]

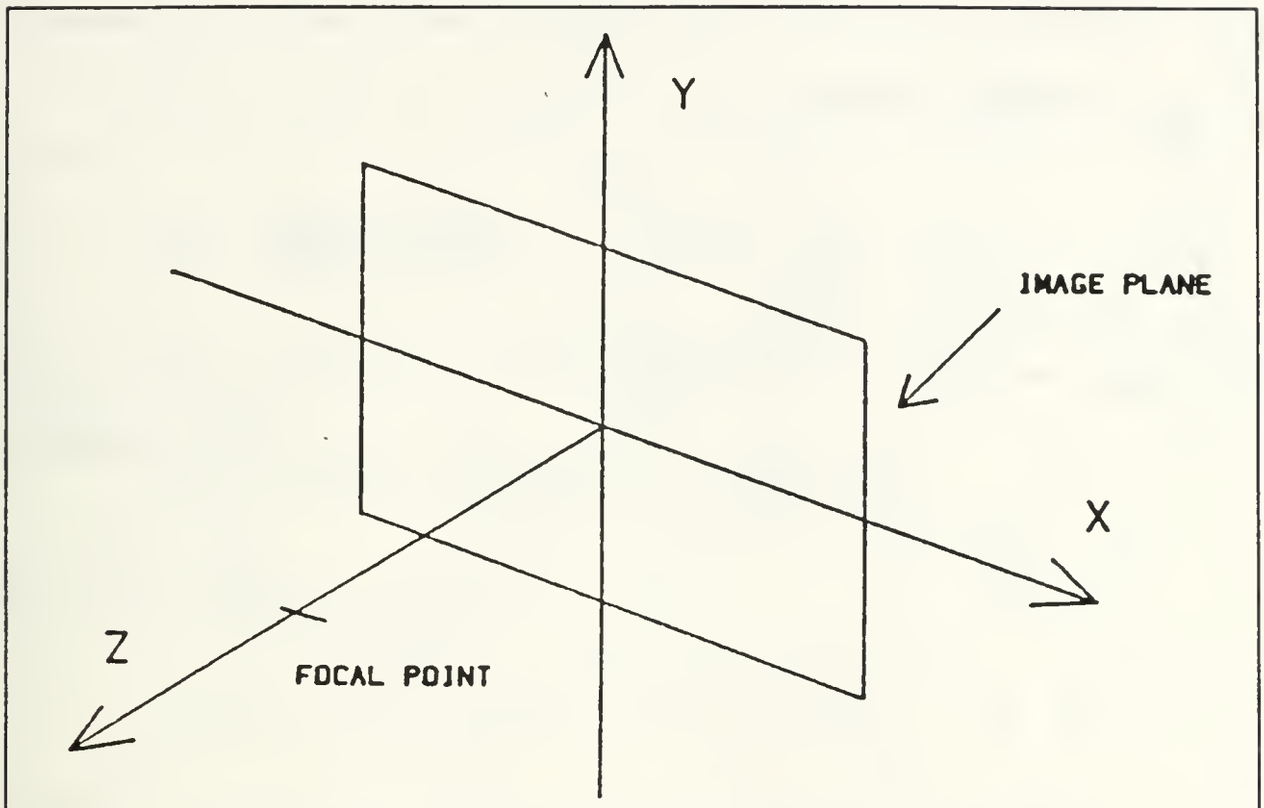


Figure 2-3 Image Coordinate System

$$M = \begin{bmatrix} \cos xX & \cos xY & \cos xZ \\ \cos yX & \cos yY & \cos yZ \\ \cos zX & \cos zY & \cos zZ \end{bmatrix} = \begin{bmatrix} M_{11} & M_{12} & M_{13} \\ M_{21} & M_{22} & M_{23} \\ M_{31} & M_{32} & M_{33} \end{bmatrix} \qquad (2.2)$$

By convention the [M] matrix transforms from the object system to the image system [Ref. 4, p. 139]. Using this matrix, every object point may be converted from the 3-D XYZ coordinates to the xyz image coordinates.

At this point the relation for a single object point is developed. Once defined, the relation may then be applied to all the vectors from the object to the focal point. Figure 2-4 illustrates the object and image space coordinate systems. Define the vector <FA> as a vector

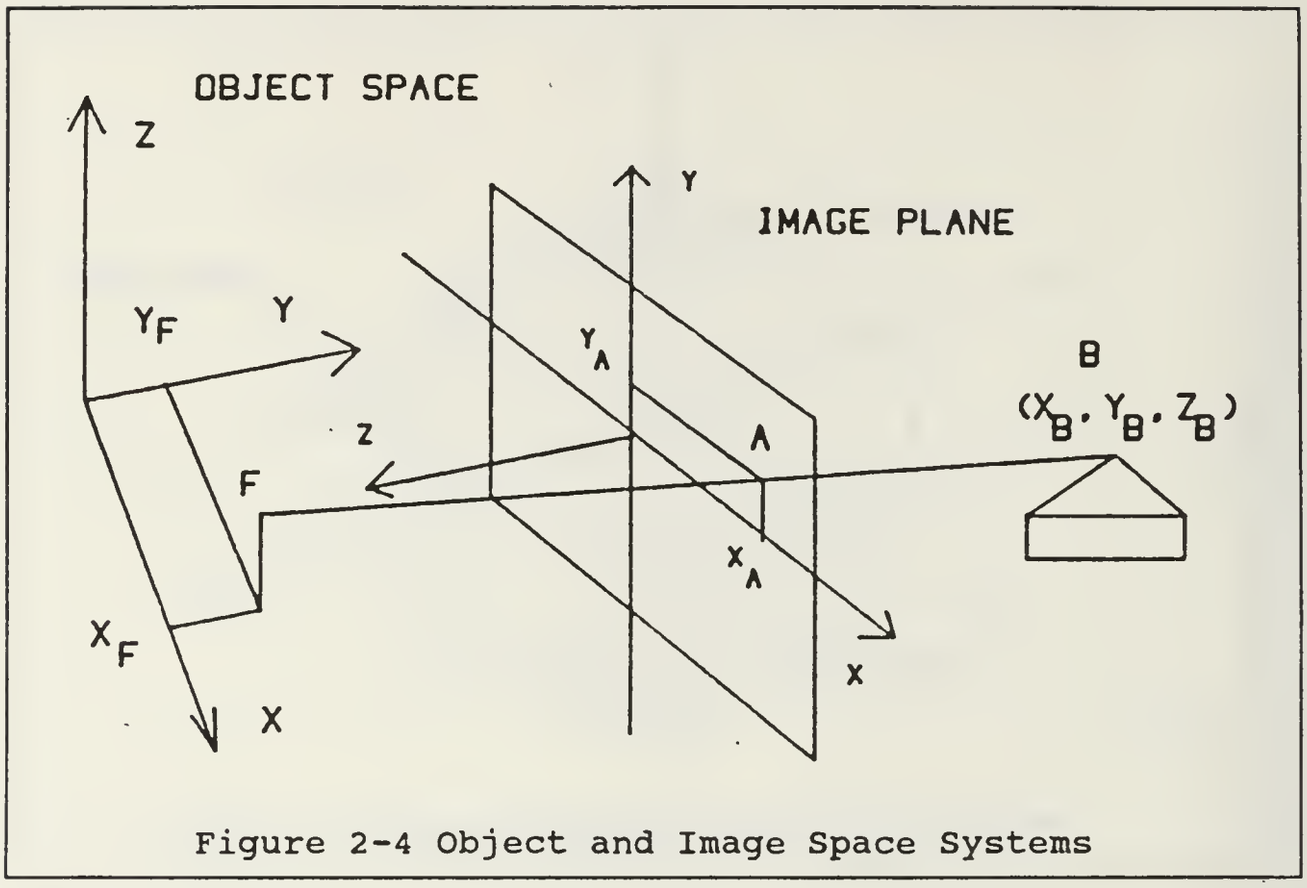


Figure 2-4 Object and Image Space Systems

from the focal point (F) to a point on the image plane (A) and the vector <FB> as a vector from the focal point (F) to an object point (B). Note that the image vector <FA> is collinear with the object vector <FB>. [Ref. 4, p. 141]

$$\overline{FA} = k\,\overline{FB}$$

where: $\overline{FA}$ - image vector

$\overline{FB}$ - object vector (2.3)

k - constant of proportionality

Using the image space coordinates (x,y,z) to describe <FA> and object space coordinates (X,Y,Z) to describe <FB> gives:

$$\overline{FA} = \begin{bmatrix} x_A - x_\emptyset \\ y_A - y_\emptyset \\ -f \end{bmatrix} \qquad \overline{FB} = \begin{bmatrix} X_B - X_F \\ Y_B - Y_F \\ Z_B - Z_F \end{bmatrix} \qquad (2.4)$$

Then using equation 2.1 to express <FA> in the xyz image system yields:

$$\overline{FA} = k \begin{bmatrix} M \end{bmatrix} \overline{FB} \qquad (2.5)$$

or after substitution:

$$\begin{bmatrix} x_A - x_\emptyset \\ y_A - y_\emptyset \\ -f \end{bmatrix} = k \begin{bmatrix} M \end{bmatrix} \begin{bmatrix} X_B - X_F \\ Y_B - Y_F \\ Z_B - Z_F \end{bmatrix} \qquad (2.6)$$

Equation 2.6 is a form of the collinearity equation, it forms the basic relationship that the focal point, the image point and the object point are in a straight line. The values of x0 and y0 allow for the observer to be slightly misaligned with the image plane z-axis and generally are set equal to zero. This is the fundamental relationship used in this project to create the perspective views on the monitor screen. [Ref. 4, p. 141] This transformation will be called upon after the elevation points have been grouped into triangles and the observer location has been entered. Each elevation point in front of the observer will be transformed to image plane coordinates for subsequent drawing on the 2-D screen.

III. DATA FILE FORMAT FOR DIFFERENT SENSOR IMAGES

Input to the routines consist of two data files, the bathymetric data and the image data. Each of these is described below.

A. BATHYMETRIC DATA

The bathymetric data was obtained from the Defense Mapping Agency/National Geophysical Data Center. The complete data base should be referenced as "Digital Bathymetric Data Base-Unclassified" or (DBDBU). The area which was used here extends from 50 degrees north latitude, 140 west longitude to 32 degrees north, 120 west on a 5 minute by 5 minute grid. Each value describes the depth in meters below sea level at a given coordinate location. Values which lie above sea level are assigned -10 to avoid ambiguity.

Extracting the area of interest is performed by a utility program, CROPELEV.FOR. The user enters the latitude and longitude of the southwest corner of the area of interest and the number of rows and columns desired. Each row and column is 5 minutes of latitude and longitude respectively.

After the area of interest is selected, the data is extracted from the original file and placed in row-column format where the rows correspond to latitude and the columns represent the longitude. Row numbers increase from south to north and column numbers increase from west to east. The first record of the file is a header containing the latitude and longitude (deg, min, sec) of the southwest corner, the latitude and longitude resolution in seconds and the number of rows and columns of data. Figure 3-1 illustrates this file format. This file is stored on disk and loaded into the variable RELEV(*,*) at runtime.

B. IMAGERY DATA

The sonar imagery data was obtained from the U.S. Geological Survey, Department of the Interior "Atlas of the Exclusive Economic Zone, Western Conterminous United States", [Ref. 5]. This atlas consists of 36 two degree mosaic images at a scale of 1:500,000. Coverage extends from 49 degrees north, 130 west to 30 degrees north, 117 west. These images were obtained using a unique side-scan sonar system called GLORIA (Geological Long-Range Inclined Asdic). [Ref. 5, p. 2] This sonar system allows mapping of large areas of ocean bottom on a single pass of the ship. Figure 3-2 lists several characteristics of the GLORIA system.

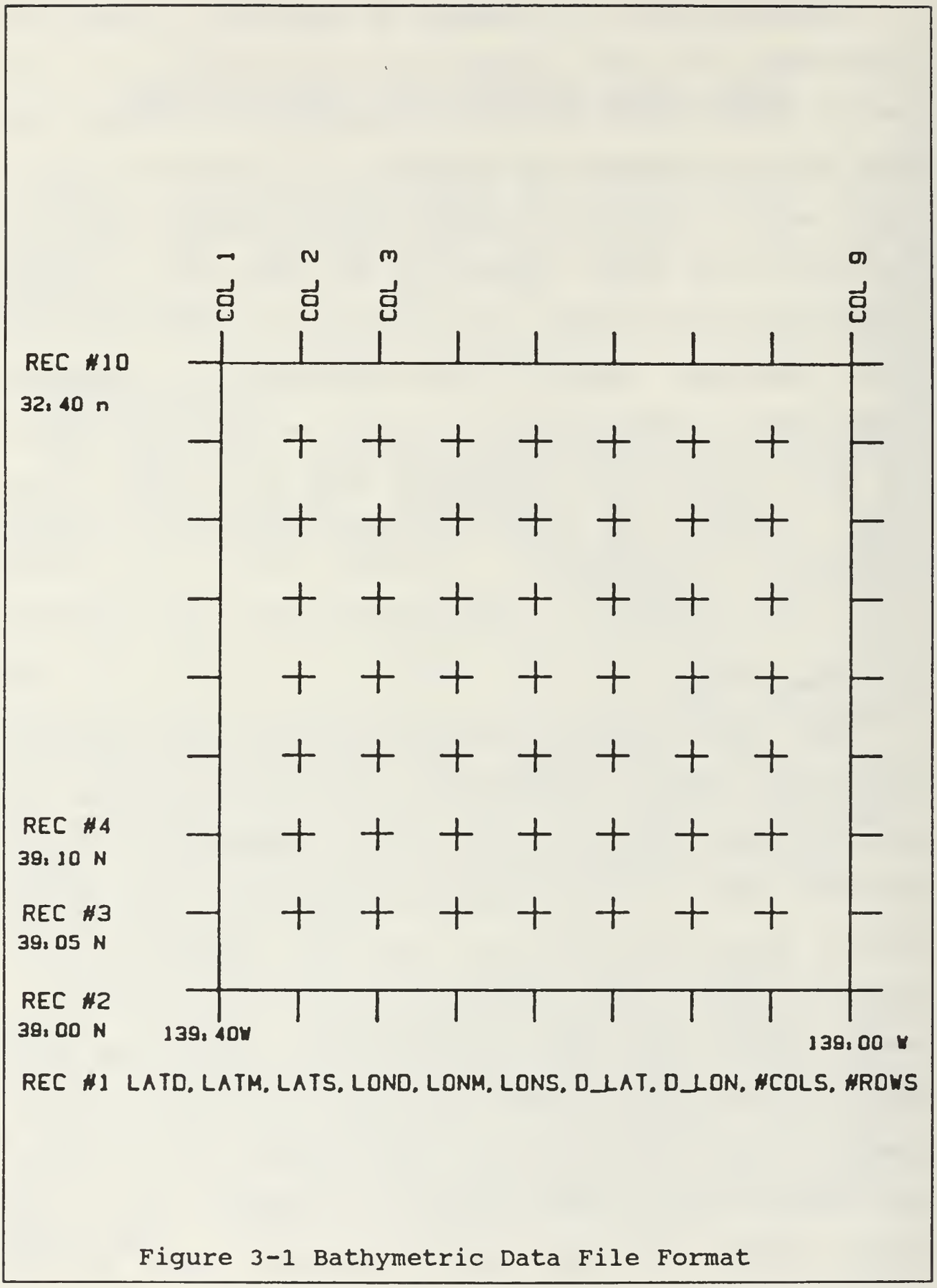


Figure 3-1 Bathymetric Data File Format

Gloria Side Scan Sonar Specifications

Size	7.75 x .66 meters
Weight	2 tons
Scan	30 km/side = 60 km swathwidth at 10 kts.
Power	10 kw/side
Beamwidth	2.5 deg azimuth 10 deg vertical
Resolution	30 x 218 meters/pixel at 5000 meters depth
Frequency	6.5 kHz

Figure 3-2 Some Characteristics of the Gloria II Side-Scan Sonar [Ref. 6, p. 7]

Each mosaic image is a half-tone black and white print of the acoustic reflectance of the sea floor with white representing the highest reflectance and black the lowest reflectance. As seen in Figure 3-2, the resolution of each pixel is distorted (30 meters by 218 meters) due to the ships motion along a track perpendicular to the scanning direction. The smaller value (30 meters) is the resolution in the cross-track direction while the larger value (218 meters) is in the along track direction. Prior to forming the mosaic images, aspect ratio distortion is removed by correcting for geometric and radiometric

distortions in the raw data. The final resolution, after correction, used to create the mosaics is about 50 meters x 50 meters. This relatively low resolution image can show only large scale features but may point out areas which deserve additional attention. Also note that this resolution is several orders of magnitude better than the available bathymetric data. [Ref. 6, p. 3]

Digital sonar imagery data was not available for use in this thesis. In order to determine the effectiveness of this imaging method, the digital image data was created by locally digitizing the mosaics contained in the atlas. This provided digital imagery data in the following form, 512 x 512 pixels with each pixel using one byte of storage resulting in 256 shades of gray. This data is stored on disk in direct access form as 512 fixed length records with 512 bytes per record. Record one is the northern end of the image and record 512 is the most southern end of the image. Figure 3-3 illustrates the orientation of the image file. The disk file is loaded into the variable IMAGE(*,*) at runtime.

Ideally, the digital imagery data would be directly available, either from the GLORIA side-scan sonar or from other sonar systems. If this were the case, the data would require processing to remove the different sources of error and to convert it from its native form into the form required in this project. Processing techniques for

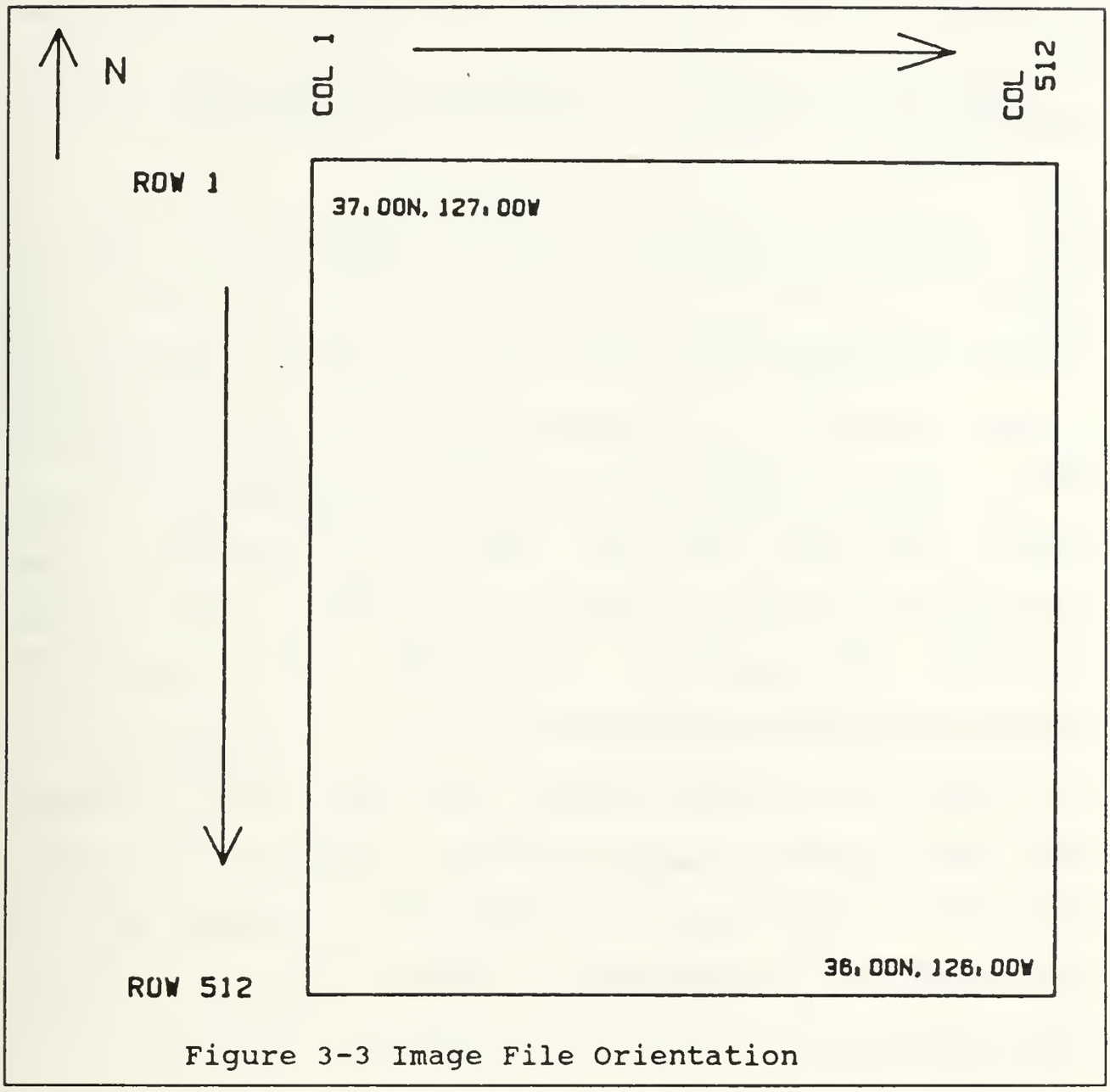


Figure 3-3 Image File Orientation

GLORIA digital data is discussed in Ref. 7, "Processing Techniques for Digital Sonar Images from GLORIA" by Pat S. Chavez, Jr.

The current limitation for input data is 70 rows by 70 columns for the elevation data and 1024 rows by 1024 columns for the image data. These dimensions may be increased to accommodate higher resolution data; the

tradeoff is speed of execution. The number of triangular panels is a function of the elevation data size, specifically:

$$\#panels = ((\#rows - 1) * 2) * (\#cols - 1) \qquad (3.1)$$

An elevation file with resolution 20 x 20 has 722 panels; increasing the resolution to 100 x 100 results in 19,602 panels. Each panel is projected point by point then sorted and drawn for every image view constructed. The increase in computation time follows directly. In addition to the elevation file resolution, the image file resolution may be increased. The cost in computation is not severe in this case because the image file is accessed only once when the image pixels are averaged to calculate the color or gray level for the individual panels. This is done prior to displaying the first image and is not repeated unless the program is exited and restarted.

IV. ALGORITHMS FOR COMBINED 3-D PERSPECTIVE DISPLAY

The main routine begins by reading two input data files into the arrays RELEV(*,*) and IMAGE(*,*). Figure 4-1 illustrates the steps involved in displaying the image on the monitor.

The procedure begins by forming the triangles in the grid of the elevation data, averaging the image pixels inside each triangle and assigning the average gray intensity to the panel (one triangle). The observer location is read, then the image plane coordinate system is constructed. Based on the orientation of the image plane the [M] matrix is calculated and used to map the elevation points to the image plane. A 2-D to 2-D affine transform is used to convert the points from image plane coordinates to the screen coordinates. To display the perspective view on a flat screen, triangles which are hidden behind foreground objects must be removed. This hidden surface removal is performed by calculating the distance from the panel to the observer and drawing the panels in sequence from the farthest to the nearest. This chapter will cover these areas in detail and also discuss alternatives which were considered, but not implemented.

```
Begin

    Load input data file
        Read in elevation (bathymetric) data
        Read in imagery data

    Construct triangles in elevation data grid
        Average the gray shade in each triangle

    Get observer location

    Form the image plane vectors
        Calculate the elements of the [M] matrix

    Transform to the image plane
        Determine which panels are visible

    Convert to screen coordinates

    Remove hidden surfaces

    Plot results on screen

End
```

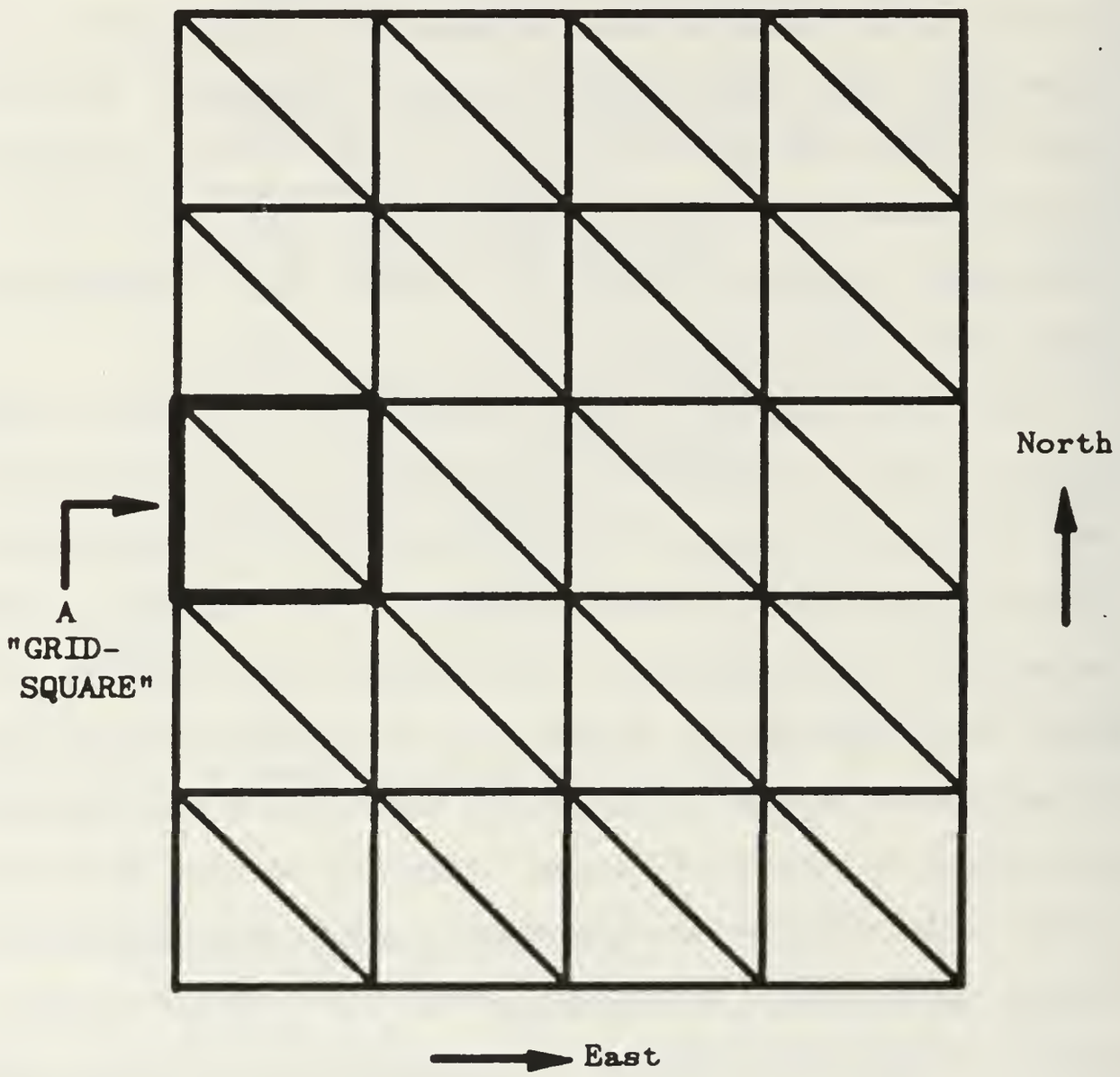
Figure 4-1 Basic Program Flow

A. POLYGON FORMATION AND SHADING

Solid objects may be represented in numerous ways. Some objects lend themselves to being described in terms of a number of planes or surfaces. A cube, for example may be precisely defined with six planes [Ref. 8, p. 189]. As the object or scene becomes more complex, the number of surfaces required to accurately describe it increases. This method of representing a 3-D surface by plane surfaces lends itself particularly well to this

application. It allows for easy calculation of the color or gray level by averaging of pixels contained within the panel, easy transformation to the image plane, and easy use of graphics hardware to execute the polygon fill operation. This last feature is most important; earlier work in this area pointed out the amount of time required to draw based on a point by point method [Ref. 2, p. 56]. Using the graphics hardware polygon fill capability alleviates this problem.

The input elevation data is supplied in gridded form and is easily partitioned into squares and ultimately into triangles. Figure 4-2 illustrates the partitioning of the elevation data into triangles. The procedure for selecting the gridsquares and forming the triangles is given as psuedocode in Figure 4-3. The column coordinates of each point are stored in IA(*) and the row coordinates are stored in the JA(*) array. Starting in the southwest corner, the program selects four elevation points which form a gridsquare. These four points are called node_a, node_b, node_c and node_d. The gridsquare is divided into two triangles by calculating the equation of the line which divides it. This dividing line is completely described by the slope and y-intercept. The column boundaries are defined by IA(node_a) and IA(node_b); the row boundaries are defined by JA(node_a) and JA(node_c). For each incremental step from the eastern (right hand)



View from above looking down on the terrain.

-Terrain elevation points are connected to form triangular polygons with common edges.

Figure 4-2 Polygonal Terrain Construction
[Ref. 3, p. 35]

```
BEGIN
  DO   select a gridsquare
     (*select four nodes which make one gridsquare*)
     READ node_a
     READ node_b
     READ node_c
     READ node_d
     (*calc slope and y-intercept of line forming
     the triangle*)
     slope = rise / run
     y-intercept = y - slope * (x)
     DO M = IA(node_b) to IA(node_a)  step -1
        IY=(slope) * M + y-intercept
        DO L = JA(node_b) to IY step -1
           (*average image pixels in lower triangle*)
        END DO
        DO L = IY to JA(node_c) step -1
           (*average image pixels in upper triangle*)
        END DO
     END DO
     PL_AR(*,*) = nodes of triangle, average gray shade
  END DO
END
```

Figure 4-3 Polygon Formation Psuedocode

boundary, IA(node_b) to the western (left hand) boundary, IA(node_a), a value of y equal to IY is calculated based on the equation of the dividing line. The image pixels above and below this line are averaged separately for the upper and lower triangles. The process is repeated for each incremental step in the columns from right to left. This scanning and averaging pattern is illustrated in Figure 4-4. Note that the scan pattern is from bottom to top, right to left in each triangle. [Ref. 2, p. 39] The average shade of the triangle along with the panel number and vertices are stored in array PL_AR(*,*). The data structure is shown in Figure 4-5. Notice that PL_AR(*,5) generally is not used, it will be used to store the maximum distance of the plane from the observer.

B. IMAGE PLANE FORMATION, PERSPECTIVE VIEW CALCULATION

After the shading for each panel is calculated, the observer's location is entered. This consists of latitude and longitude (deg, min, sec), height with respect to sea level and heading. The latitude, longitude and height data is converted to object space coordinates X1,Y1,Z1 while the heading is used to select a second point in front of the observer. This second point X2,Y2,Z2 forms the line of sight (LOS) vector for the observer. The negative LOS

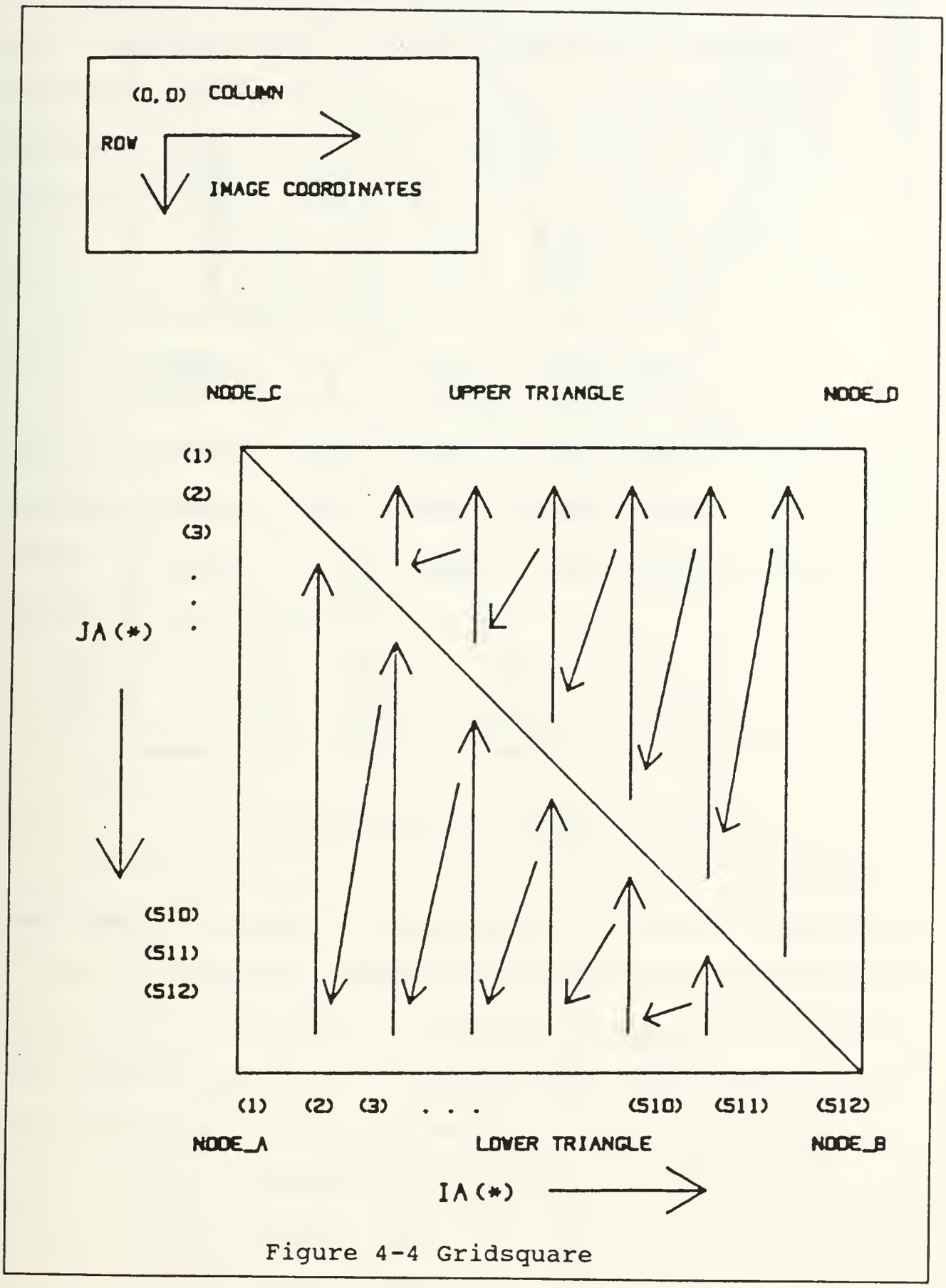


Figure 4-4 Gridsquare

TRIANGLE IDENTIFIER	NODE_A FIRST VERTEX I.D.	NODE_B SECOND VERTEX I.D.	NODE_C THIRD VERTEX I.D.	GRAY SHADE	MAXIMUM DEPTH
1	3660	3721	3720	142	109979
2	3659	3720	3719	122	109178
3	3659	3660	3720	123	109178
4	3599	3660	3659	117	108434
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.

Figure 4-5 Layout of PL_AR Array

vector is the z-axis of the image coordinate system as seen in Figure 2-3. This is formulated as shown:

$$\begin{aligned} ZVEC_X &= X1 - X2 \\ ZVEC_Y &= Y1 - Y2 \\ ZVEC_Z &= Z1 - Z2 \end{aligned} \tag{4.1}$$

$$\overline{ZVEC} = ZVEC_X\,\hat{x} + ZVEC_Y\,\hat{y} + ZVEC_Z\,\hat{z}$$

The image plane y-axis is constructed by creating a vector which points upward from the center of the earth to the observer location. The (X,Y,Z) coordinates of the observer location (X1,Y1,Z1) form this vector. The y-axis is described by:

$$\begin{aligned} YVEC_X &= X1 \\ YVEC_Y &= Y1 \\ YVEC_Z &= Z1 \end{aligned} \qquad (4.2)$$

$$\overline{YVEC} = YVEC_X\,\hat{x} + YVEC_Y\,\hat{y} + YVEC_Z\,\hat{z}$$

The image coordinate system is a right-hand system. Therefore, the cross product of the y-axis and z-axis forms the x-axis. The x-axis of the image plane is constructed as follows:

$$\begin{aligned} XVEC_X &= ((YVEC_Y \times ZVEC_Z) - (YVEC_Z \times ZVEC_Y)) \\ XVEC_Y &= ((YVEC_Z \times ZVEC_X) - (YVEC_X \times ZVEC_Z)) \\ XVEC_Z &= ((YVEC_X \times ZVEC_Y) - (YVEC_Y \times ZVEC_X)) \end{aligned} \qquad (4.3)$$

$$\overline{XVEC} = XVEC_X\,\hat{x} + XVEC_Y\,\hat{y} + XVEC_Z\,\hat{z}$$

The [M] matrix is calculated using the relationship developed in Chapter Two, equation 2.2. Recall that each element of the [M] matrix is the cosine of the spatial angle between the axis of the image coordinate system and the object coordinate system axis. The object space coordinate axis may be represented as unit vectors:

$$\begin{aligned} \overline{XAXIS} &= 1\hat{x} + 0\hat{y} + 0\hat{z} \\ \overline{YAXIS} &= 0\hat{x} + 1\hat{y} + 0\hat{z} \\ \overline{ZAXIS} &= 0\hat{x} + 0\hat{y} + 1\hat{z} \end{aligned} \qquad (4.4)$$

The cosine of the spatial angle between two vectors is determined from the dot product of the two vectors.

$$\bar{A} \cdot \bar{B} = |A||B| \cos \theta_{AB} \qquad (4.5)$$

The dot product may be expanded as shown:

$$\cos \theta_{AB} = \frac{\bar{A} \cdot \bar{B}}{|A|\,|B|}$$

$$\cos \theta_{AB} = \frac{A_X B_X + A_Y B_Y + A_Z B_Z}{|A|\,|B|} \qquad (4.6)$$

Substituting the image space x-axis vector (xvec) and the object space X-axis vector (XAXIS) for [A] and [B] allows M11 to be calculated:

$$M_{11} = \cos \theta_{xX} = \frac{XVEC_X \cdot XAXIS_X + XVEC_Y \cdot XAXIS_Y + XVEC_Z \cdot XAXIS_Z}{|XVEC|\;\;|XAXIS|}$$

$$= \frac{(XVEC_X \cdot 1) + (XVEC_Y \cdot 0) + (XVEC_Z \cdot 0)}{|XVEC| \cdot 1} \qquad (4.7)$$

$$M_{11} = \frac{XVEC_X}{|XVEC|}$$

The other elements of the [M] matrix may be found in a similar fashion. This results in:

$$M_{11} = \frac{XVEC_X}{|XVEC|} \qquad M_{12} = \frac{XVEC_Y}{|XVEC|} \qquad M_{13} = \frac{XVEC_Z}{|XVEC|}$$

$$M_{21} = \frac{YVEC_X}{|YVEC|} \qquad M_{22} = \frac{YVEC_Y}{|YVEC|} \qquad M_{23} = \frac{YVEC_Z}{|YVEC|} \qquad (4.8)$$

$$M_{31} = \frac{ZVEC_X}{|ZVEC|} \qquad M_{32} = \frac{ZVEC_Y}{|ZVEC|} \qquad M_{33} = \frac{ZVEC_Z}{|ZVEC|}$$

This method of calculating the image coordinate axis and the corresponding [M] matrix elements results in a viewing plane which is oriented vertically and faces in the direction of the observer course.

An alternative to this method is that, instead of specifying the course to see the perspective view, one can have the image plane always face the object. This would simulate rotating the object while keeping the observer location fixed. The decision to use the first method is based on the feeling that it results in a more natural view for the observer.

C. TRANSFORM TO THE IMAGE PLANE

After the [M] matrix is constructed the elevation points are projected onto the image plane. Points which lie behind the image plane or behind the observer are not projected. The method used to select which points to project and the projection equations are presented next.

Determining which elevation points are located behind the image plane is done by finding the cosine of the angle between a vector formed by the negative z-axis of the image plane and a vector drawn from X2,Y2,Z2 to the object point. Solving for the cosine of the angle and not the angle itself eliminates the need to use trigonometric functions, and provides faster execution. Psuedocode for this procedure is given in Figure 4-6.

```
BEGIN

    DO  IR = 1 to Number of elevation pts.

        (*get XYZ coordinates of point*)

        X = XYZ(IR,1)
        Y = XYZ(IR,2)
        Z = XYZ(IR,3)

        (*form the observer LOS vector*)

        OBS_VECX = X2 - X1
        OBS_VECY = Y2 - Y1
        OBS_VECZ = Z2 - Z1

        (*form the object vector*)

        OBJ_VECX = X - X2
        OBJ_VECY = Y - Y2
        OBJ_VECZ = Z - Z2

        (*find the cosine of the angle between the
        OBS_VEC and OBJ_VEC*)

        COS_THETA = (dot product of OBS_VEC, OBJ_VEC)/
                        (Mag(OBS_VEC) x Mag(OBJ_VEC))

        IF (COS_THETA > 0) THEN
            Project onto image plane
            Calculate depth of panel
        ELSE
            Mark as a non-visible point
        END IF

    END DO

END
```

Figure 4-6 Psuedocode Determine Which Panels to Project

The procedure begins by forming two vectors, the observer LOS vector and the observer to object vector (object vector). Ideally, the origin of the image plane coordinate system would be used to form the "observer to object vector", but the (X,Y,Z) coordinates of this point are not readily known. The observer location (X1,Y1,Z1) is not used because it is behind the image plane and will result in some panels being viewed which are behind the image plane. The point (X2,Y2,Z2) is used as an approximation of the origin of the image plane. If the angle between the LOS vector and the object vector is greater than 90 degrees, the point lies behind the image plane and will not be projected onto the image plane. Figure 4-7 illustrates the orientation of the vectors and the image plane. The cosine of the angle between the two vectors is again found by using the dot product method.

If the angle is less than 90 degrees, the cosine will be a positive number. If the angle is greater than 90 degrees the cosine will be negative; this indicates the point is not in the hemisphere in front of the observer. If the point is not visible in the forward hemisphere then the point is flagged as hidden. Generally each point is associated with up to six triangles; marking each point as hidden will prevent attempting to draw triangles which may be partially behind the image plane.

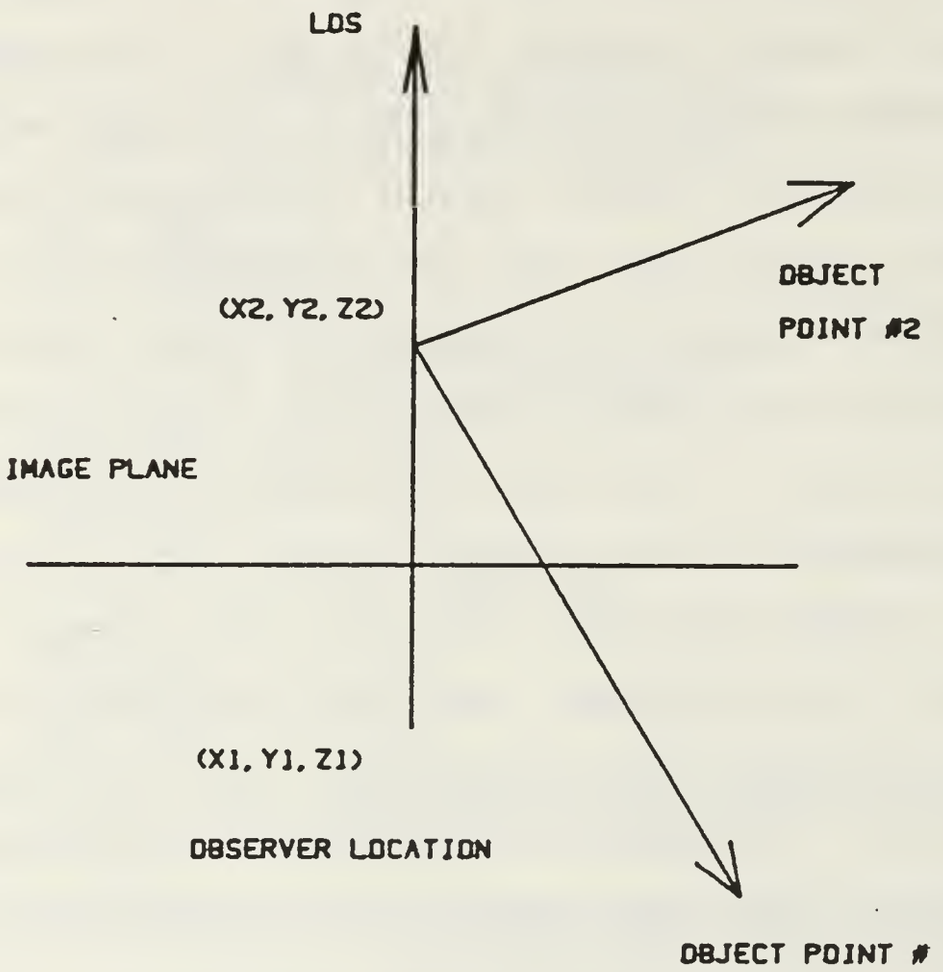


Figure 4-7 Image Plane Object Vector

The points which are in the forward hemisphere are projected onto the image plane using the relation developed in Chapter Two, equation 2.6 which is repeated here for clarity:

$$\begin{bmatrix} x_A - x_\emptyset \\ y_A - y_\emptyset \\ -f \end{bmatrix} = k \begin{bmatrix} M \end{bmatrix} \begin{bmatrix} X_B - X_F \\ Y_B - Y_F \\ Z_B - Z_F \end{bmatrix} \qquad (4.9)$$

Expanding to three separate equations gives:

$$x_A - x_\emptyset = k\left[M_{11}(X_B-X_F) + M_{12}(Y_B-Y_F) + M_{13}(Z_B-Z_F) \right] \quad (4.10)$$

$$y_A - y_\emptyset = k\left[M_{21}(X_B-X_F) + M_{22}(Y_B-Y_F) + M_{23}(Z_B-Z_F) \right] \quad (4.11)$$

$$-f = k\left[M_{31}(X_B-X_F) + M_{32}(Y_B-Y_F) + M_{33}(Z_B-Z_F) \right] \quad (4.12)$$

Then dividing (4.10) and (4.11) by (4.12) yields:

$$x - x_\emptyset = -f\left[\frac{M_{11}(X_B-X_F) + M_{12}(Y_B-Y_F) + M_{13}(Z_B-Z_F)}{M_{31}(X_B-X_F) + M_{32}(Y_B-Y_F) + M_{33}(Z_B-Z_F)}\right] \quad (4.13)$$

$$y - y_\emptyset = -f\left[\frac{M_{21}(X_B-X_F) + M_{22}(Y_B-Y_F) + M_{23}(Z_B-Z_F)}{M_{31}(X_B-X_F) + M_{32}(Y_B-Y_F) + M_{33}(Z_B-Z_F)}\right] \quad (4.14)$$

Equations (4.13) and (4.14) are the relations used to transform the elevation data points to the image plane. They allow projecting all elevation points in the hemisphere forward of the observer at one time. Altering the size of the image plane will not require the elevation points to be projected a second time. [Ref. 4, p. 142]

D. AFFINE IMAGE PLANE TO SCREEN TRANSFORM

The image plane coordinates must be transformed to the screen coordinate system. Figure 4-8 illustrates the image plane and the screen coordinate systems. The maximum frame size values of the x and y axis in the image plane system are determined by the desired magnification or field of view and the y-scale factors.

In general, the affine transform will map between any 2-D coordinate systems allowing for a rotation of the

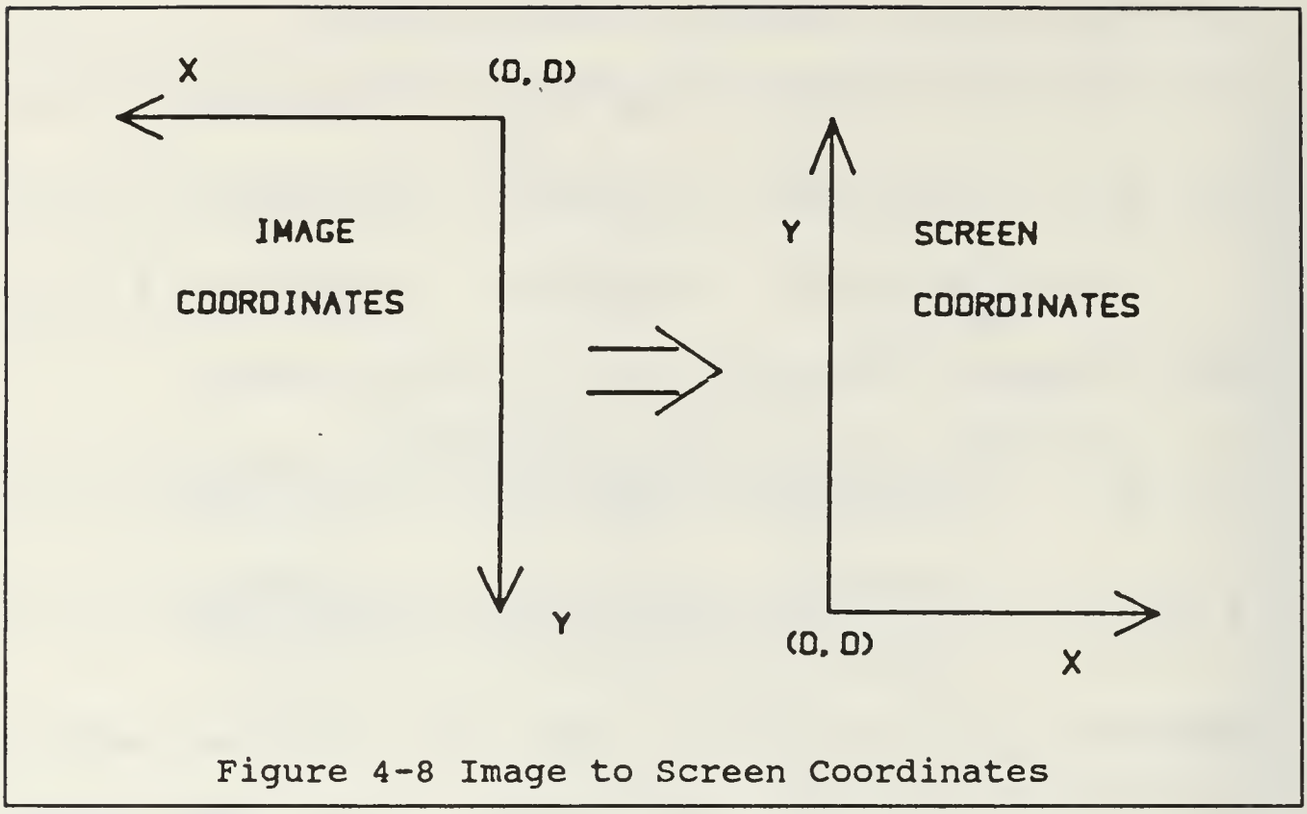


Figure 4-8 Image to Screen Coordinates

axis, horizontal and vertical scale changes, two translations and a non-perpendicularity of the axis. The general form of the affine transform is [Ref. 4, p. 593]:

$$\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \frac{1}{a_1 b_2 - a_2 b_1} \begin{bmatrix} b_2 & -b_1 \\ -a_2 & a_1 \end{bmatrix} \begin{bmatrix} x_2 - C_1 \\ y_2 - C_2 \end{bmatrix} \qquad (4.15)$$

where: $a_1 = S_x (\cos \beta - \xi \sin \beta)$

$a_2 = S_x (\sin \beta + \xi \cos \beta)$

$b_1 = S_y (-\sin \beta)$

$b_2 = S_y (\cos \beta)$

ξ represents a non-perpendicularity of the axis

β represents a rotation of the axis

C_1 = x translation of origin

C_2 = y translation of origin

Starting with a magnification factor of 1.0 and a y-scale factor of 1.2, the x and y frame sizes are 35000 and 29000 respectively. These values must be mapped onto a screen coordinate system which ranges from 1-512 in both the x and y direction. There are two scale changes, two translations, no rotations and no non-perpendicularities involved with this transformation. This results in sigma equal to zero and beta equal to zero. Substituting these values results in:

$$a_1 = S_x \qquad b_1 = \emptyset$$
$$a_2 = \emptyset \qquad b_2 = S_y \tag{4.16}$$

Inserting these values and expanding the matrix equations gives:

$$x_1 = \frac{x_2 - C_1}{S_x} \qquad y_1 = \frac{y_2 - C_2}{S_y} \tag{4.17}$$

The scale factors are formed at runtime to allow the changes in magnification (field of view) and elevation scale. Figure 4-9 is psuedocode which implements the 2-D transform. Note that the variables A1, A2, B1, B2 are in terms of the image plane frame size. This approach allows the user to change the magnification or elevation scaling interactively at this point. Also note that each elevation point is checked for non-visible status prior to the transform. This checking saves execution time by avoiding points not in the field of view.

```
BEGIN

    C1 = image x axis maximum

    C2 = image y axis maximum

    A1 = image x axis frame size / screen x axis maximum

    B2 = image y axis frame size / screen y axis maximum

    DO  IR = 1 to Number of elevation pts.

       IF (point .EQ. visible) THEN
          xscreen = (ximag - C1) / A1
          yscreen = (yimag - C2) / B2
       END IF

    END DO

END
```

Figure 4-9 Affine Transform Psuedocode

E. HIDDEN SURFACE REMOVAL

In order to properly display the perspective view, the surfaces hidden behind front surfaces must be dealt with. Earlier work utilized a Z-buffer to perform the hidden surface removal [Ref. 2, p. 41]. The Z-buffer method utilizes a pixel by pixel depth buffer. Each pixel position on the screen corresponds to a location in the buffer. The buffer is initialized to a maximum depth, then prior to drawing each pixel the depth is compared with the depth already stored in the buffer. If the pixel depth is less than the depth stored in the buffer, then the pixel

is drawn and the new depth is stored in the depth buffer. Earlier work by Coleman pointed out the poor performance of this technique when executed in software [Ref. 2, p. 55]. Several other methods of hidden surface removal were investigated; each will be discussed briefly and the disadvantages listed.

First, the method of "bounding rectangles" was investigated. In this method a rectangle is formed around each triangle. This rectangle is as small as possible and is constructed from the three vertices forming the triangle. The (x,y) coordinates of each vertex are checked and the minimum and maximums of both the x and y coordinates form the boundaries of the rectangle. Now the vertices of all other triangles are compared to the rectangle. If the (x,y) coordinates of a vertex lie within the boundary of the rectangle, the triangle is marked as overlapping and the Z-buffer routine is used to handle the hidden surface removal. If the triangle does not overlap any other triangles, it is drawn to the screen. Originally the intent was to reduce the number of panels which needed to be drawn using the Z-buffer method. This was not achieved because the panels are triangular vice rectangular. Several experiments showed that no triangles passed the bounding rectangle test and consequently no savings in execution time was realized. [Ref. 9, p. 246]

A second method utilizes a test function for each side of a triangle. A test function is formed for each side of a triangle. This test function is formed from the equation of the line describing the side. The vertices of all other triangles are checked against the test function. This results in nine comparisons for every triangle checked. The calculation time is excessive, also the results were similar to the results of the bounding box test. Very few triangles passed the test and the Z-buffer method had to be used on the majority of panels, with no savings in execution time.[Ref. 9, p. 24]

Another approach, known as the "Painter's algorithm", is not hidden surface removal at all. [Ref. 8, p. 265] Here the panels are drawn to the screen in sequence from background to foreground. The panels which are hidden from view are covered with the panels which are closer to the observer. This is the approach used in this software. Each panel is sorted into order based on the maximum distance of the vertices from the observer location. The choice of sorting algorithms has a large affect on the efficiency of this method. Originally, the routine incorporated a simple bubble sort. The cost in performance is not noticeable with a small number of panels, but as the resolution of the bathymetric data increases, the number of panels increases as shown in equation 3.1 which is repeated here.

$$\#\ \text{panels} = ((\#\text{rows} - 1)*2) * (\#\text{cols} - 1) \qquad (4.18)$$

The poor performance of the bubble sort becomes apparent as the number of panels increases beyond 500. Two alternate sorting routines were used with this routine, the shell sort and the quick sort. Table 4-1 is a table which shows the measured performance of these sort routines. Clearly for large numbers of panels the quick sort is the most efficient. Based on these results the quick sort was selected for use in the hidden surface routine. Figure 4-11 is psuedocode which describes the hidden surface removal procedure.

TABLE 4-1 SORT PERFORMANCE

Time in seconds

# panels	Bubble	Shell	Quick
500	2	<1	<1
1000	8	<1	<1
2000	33	<1	<1
3000	80	1	1
5000	210	2	1
10000	900	5	1.5
20000	---	14	3
50000	---	56	9

```
BEGIN

  (* create sort key by counting the panels in front of
  the image plane. Depth_key(L) contains the panel ID
  number *)

    DO IR = 1 to NPLANES

       IF ( panel .NOT. behind field of view) THEN
          INCREMENT COUNT
          Depth_key(COUNT) = IR
       END IF

    END DO

    (* calc maximum depth of each visible plane *)

    DO L = 1 to COUNT

       IR = Depth_key(L)

       PL_AR(IR,5)=
           MAX(DEPTH(node_a),DEPTH(node_b),DEPTH(node_c))

    END DO

    (* call sort routine to sort the panels on the
    maximum depth*)

    CALL QUICKSORT

END
```

Figure 4-11 Psuedocode for Hidden Surface Removal by Sorting

Upon completion of the hidden surface removal routine, the screen is erased and the panels are plotted. The vertices for each panel along with the shading are stored in array PL_AR(*,*). The shading value is clipped to a range of 1-250 vice the original 1-256 due to the DEC VMS

window manager. This maintains the background color of the screen and the window attributes. The panels are read in sequence using the sort index. The (x,y) screen coordinates of each vertex and the shade are passed to the plot subroutine and the panel is drawn on the screen.

F. RESULTS

Data cropped out of the primary bathymetry data file covers an area bounded by 37:00:00 N. latitude, 126:00:00 W. longitude to 36:00:00 N. latitude, 125:00:00 W. longitude. Figure 4-12 is the digitized image of the mosaic used for the image data. The first synthesized view Figure 4-13, is from an observer location of 36:0:0 N. 125:30:0 W., 4000 meters below sea level and heading equal to 000 degrees. The y-scale exaggeration is six and the magnification factor is one. Figures 4-14 a to c are displays of the same area from different observer locations. Figures 4-14a and 4-14b are offset from the original location east and west by 25 minutes of longitude. Figure 4-14c is from the same location as the original but the height is 2000 meters below sea level. These images display the ability to "see" the ocean bottom in a new fashion. The resolution of the input elevation data was originally 5 minutes by 5 minutes. This data was interpolated in both latitude and longitude to a resolution of 1 minute by 1 minute in order to achieve a

better shading rendition. This reduces the number of pixels averaged for each triangle and improves the shading of the generated image. Figure 4-15 is an image using the original 5 minute by 5 minute resolution data. Notice that the size of the panels is larger and the shading rendition is much more coarse. In addition to affecting the shading, the resolution of the elevation data directly affects the physical shaping of objects in the terrain. This program does not utilize any method of curve fitting between adjacent elevation points and simply draws straight lines between the points. This results in objects being coarsely approximated in the synthesized view.

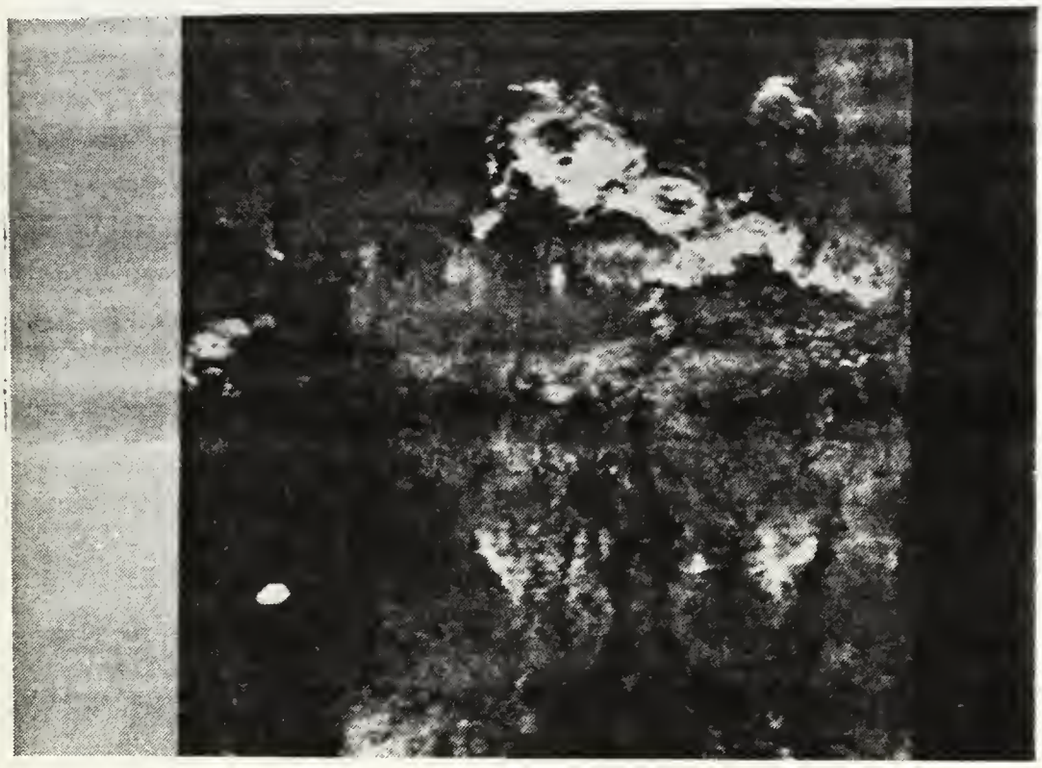


Figure 4-12 Mosaic Image

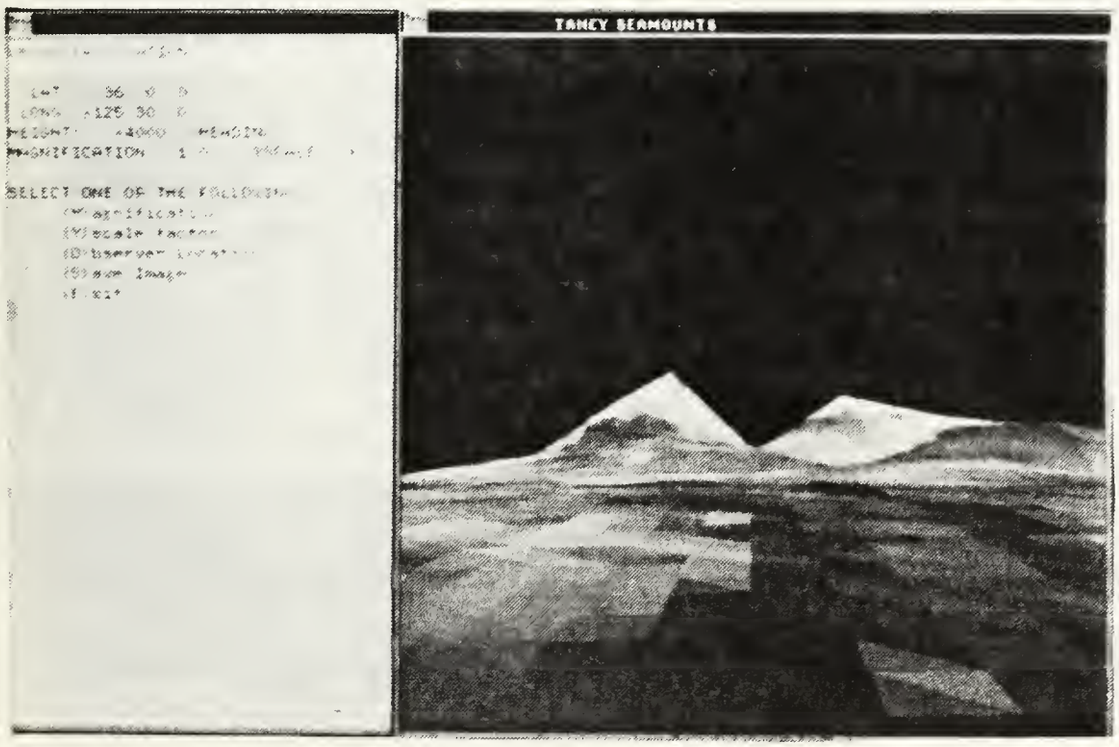


Figure 4-13 Obs Loc: 36:00:00 N Depth 4000 m
125:30:00 W Heading 000 deg

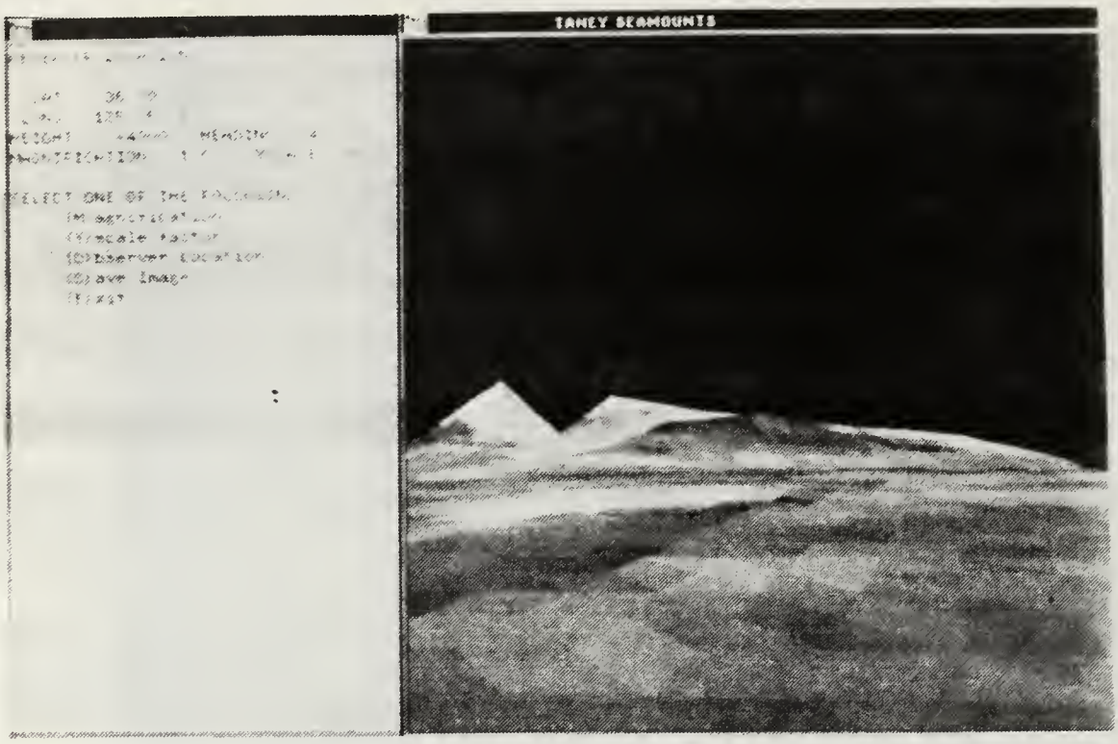


Figure 4-14a Obs Loc 36:00:00 N Depth 4000 m
125:05:00 W Heading 340 deg

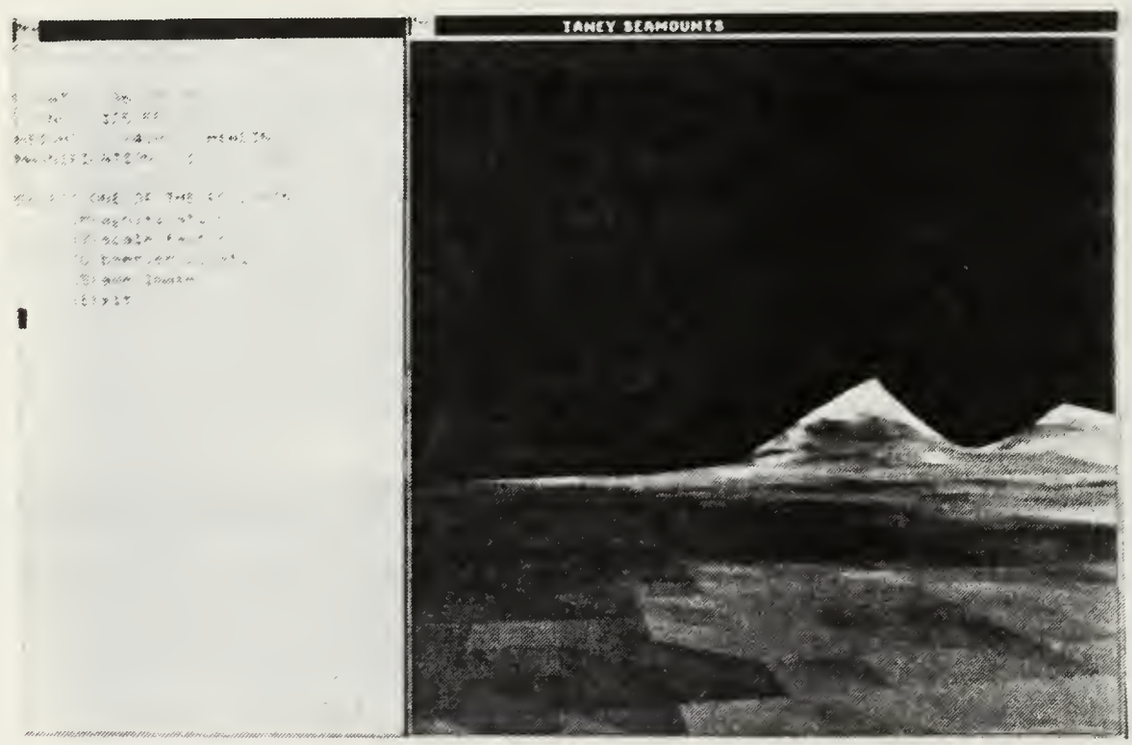


Figure 4-14b Obs Loc 36:00:00 N Depth 4000 m
125:55:00 W Heading 020 deg

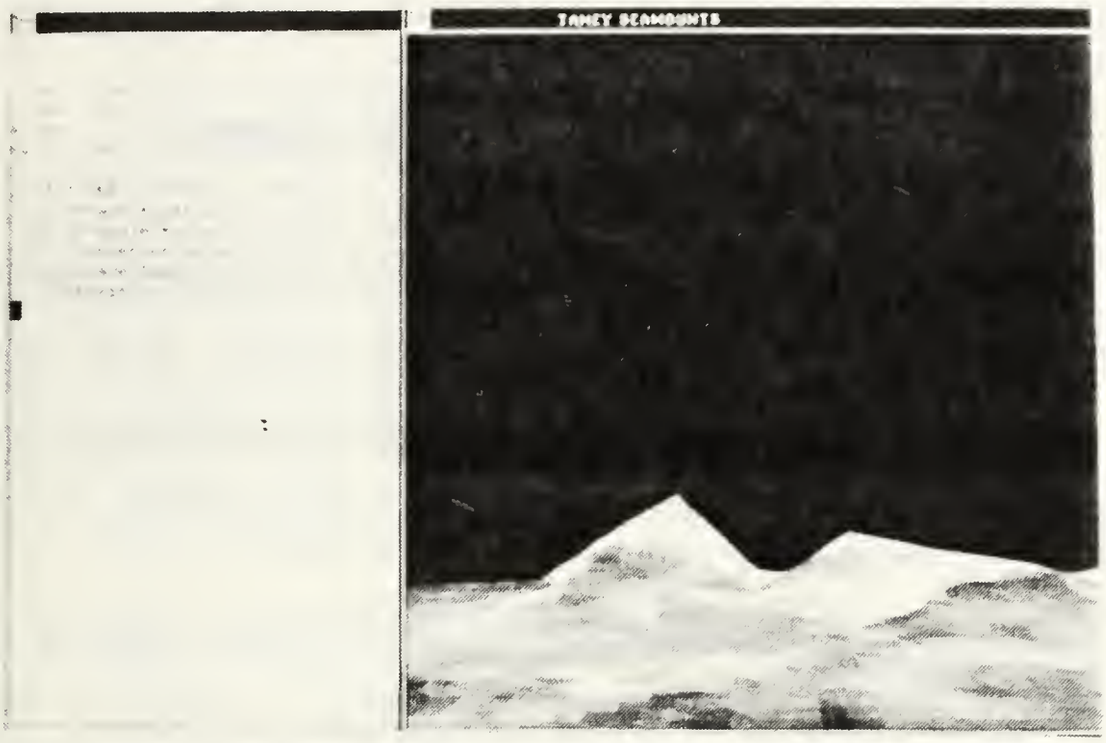


Figure 4-14c Obs Loc 36:00:00 N Depth 2000 m
125:30:00 W Heading 000 deg

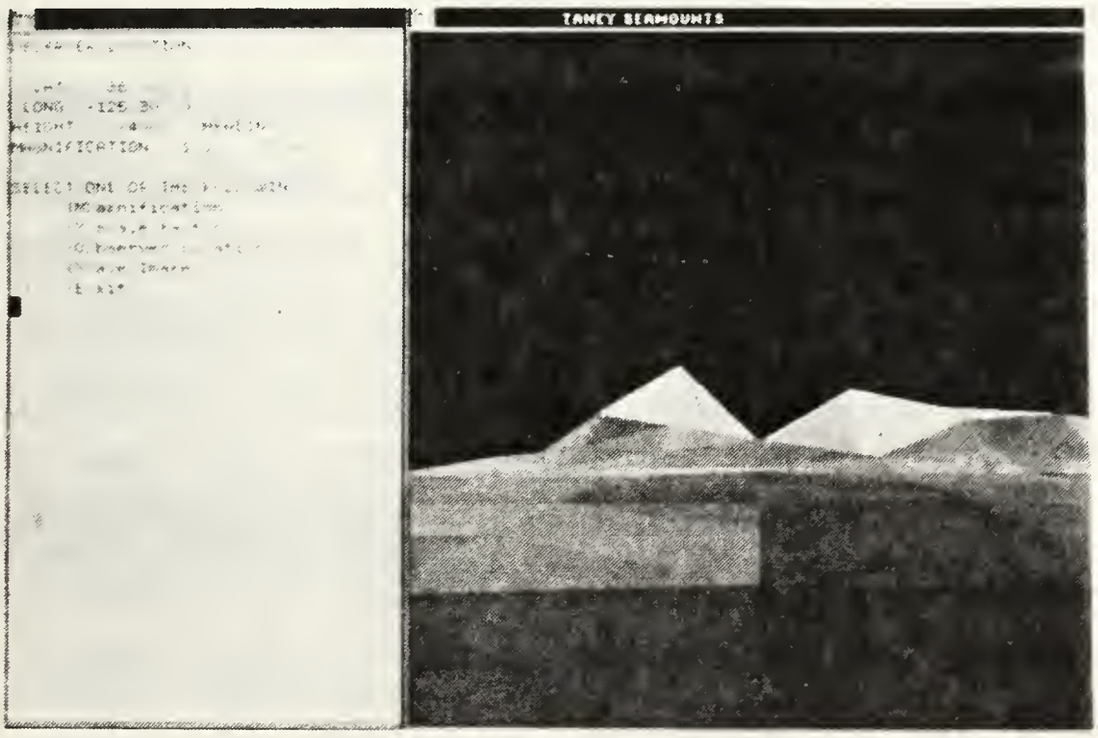


Figure 4-15 5 x 5 Minute Resolution

V. OPERATIONAL ASPECTS OF THE PROGRAM

The software utilizes an input file to load the names and geographic locations of the data files. This input file may be created by an ASCII editor or created within the program by following the prompts. The format of the input file is shown in Figure 5-1.

If the input file exists, the user simply enters the name when prompted. The program will seem to pause at this point while reading in the data files, forming the panels and averaging the image pixels. When complete with these steps, the software will prompt for the observer location. This location is entered in terms of observer latitude and longitude in degrees, minutes and seconds. Southern latitudes and western longitudes are entered as negative values. Also entered is the elevation of the observer and the heading of the observer. Elevation is in meters with respect to sea level with values greater than sea level entered as positive values and elevations below sea level entered as negative values. The heading is entered as a value from 0-360 degrees where 0 deg. is north, 90 deg. is east, 180 deg. is south, 270 deg. is west and 360 deg. is north again.

Elevation data file name

# of rows of elevation data

# of columns of elevation data

Image data file name

# of rows of image data

# of columns of image data

Latitude/Longitude of Northwest corner of image data in (deg,min,sec)

Latitude/Longitude of Southeast corner of image data in (deg,min,sec)

Title for the screen image

Figure 5-1 Input File Format

At this point the software will draw the perspective view on the screen. It takes 40 seconds to produce one image using 61 x 61 elevation points and 512 x 512 image points. After the image is drawn a menu is presented. The operator may alter the magnification or field of view, change the elevation scale exaggeration, enter a new observer location, save the image or exit the program.

A. ALTER MAGNIFICATION OR FIELD OF VIEW

The perspective view transformation simulates viewing through a camera viewfinder. The observer selects the location and points the camera, and the image is formed in

the viewfinder. The magnification or field of view is changed by changing the focal length of the lens.

The magnification factor of the initial image is one. The values of the image plane x-axis and y-axis maximums are initially set to 35000 and 29000 respectively. This loosely corresponds to a 35mm x 29mm frame size of a 35mm camera. The focal length corresponds to an initial focal length of 50mm. The field of view is given by:

$$\text{Field of View} = 2 \tan^{-1}\left[\frac{\text{xframe}}{2(\text{focus})}\right] \qquad (5.1)$$

Using the values above results in a field of view of 36.8 degrees. This field of view is approximately equal to the field of view from a 35mm camera with a 50mm "normal" lens. This field of view corresponds to a magnification factor of one. Table 5-1 shows the field of view for various magnifications.

When changing the magnification on a camera, the standard method is to change the focal length of the lens. This method presents a disadvantage here in that it requires the complete 3-D to 3-D transform to be performed for every magnification change. By changing the frame size instead of the focal length, the new image is generated faster because the change is made during the affine image plane to screen plane transformation. The magnification changes are entered as values with respect to the normal magnification.

TABLE 5-1 MAGNIFICATION VERSUS FIELD OF VIEW

Magnification	Field of View (deg)
.5	69.9
1.0	38.5
1.5	26.2
2.0	19.8
2.5	15.9
3.0	13.3
3.5	11.4
4.0	10.0
4.5	8.9
5.0	8.0

B. CHANGING THE ELEVATION SCALING

Altering the y-axis scale allows exaggeration or reduction of the y-scale. Views of the terrain with the elevation scaling set at 1:1 result in views which have very little appearance of height. By modeling the frame size as a 35mm camera frame, a 1.2:1 scaling is introduced. Research conducted by the U.S. Army Research Institute for the Behavioral and Social Sciences indicated that vertical scaling ranging from 1.25:1 to 1.50:1 presented the most natural views. [Ref. 3, p. 37]

The elevation scaling is changed in a fashion similar to the magnification as a value with respect to the 1:1 scale.

C. ENTER A NEW OBSERVER LOCATION

This option allows the operator to enter a new observer location. The current settings for magnification and elevation scaling are maintained.

D. SAVING AN IMAGE

Selecting this option allows multiple images to be saved to a disk file. When the option is selected the first time, it will prompt for the file name for the images to be stored in. Two data files are created, *.IMG and *.SIZ. If the filename extension is specified then it is used instead of the .IMG extension. Images will be saved to *.IMG file for the entire session each time the (S)ave option is selected. The entire set of images may be viewed sequentially using a program named PLAYBACK.FOR. This will display all the images stored in the image file at a rate of one image per second. Another program, UISTO512.FOR will convert a single frame from the saved images to a 512 x 512 row-column format, with 512 fixed length records and 512 bytes per record. The *.SIZ file contains the buffer size data required by the playback routine.

VI. CONCLUSIONS

A. GENERAL

The goal of this thesis was creating the 3-D perspective view of ocean bottom terrain and applying shading or color information from an image file. This goal was achieved and the ability to "see" the terrain in this fashion holds promise for future work in many fields.

The program presented here combines terrain elevation data with shading information from a second data source. This fusion of bathymetry and imagery data results in a 3-D perspective view of the ocean bottom in a fashion which appears natural to the observer. This type of display system, which combines multiple sources of data into a form that is easier to comprehend, has potential uses in mapping, geologic exploration and weapon system displays. It has direct use in any geographic information system where the data stored consists of (x,y) coordinates and an attribute. This program could merge any two layers of information from such a system and display a view which is easier to interpret. The data utilized in this project was not a part of such a system, but easily could have been. In this case the first layer of data would be the (x,y) location and the elevation with respect to sea level and the second layer of data would be the (x,y) location

and the shading value. Combining the layers of data from multiple sources is a form of sensor fusion.

B. LIMITATIONS

There are several limitations in this program. First, the image plane orientation is fixed in the vertical direction with respect to the observer heading and location. This implies that the perspective view formed simulates the view from a vehicle in level flight. Allowing the image plane to deviate from vertical would allow the observer to look down on the terrain and generate views not presently permitted. Second, the shading values are clipped to 250 vice 255 shades. This is done in order to maintain five separate entries in the virtual color map which correspond to the system colors. A third problem exists with the geographic areas of the input data. The conversion from latitude/longitude coordinates to geo-rectangular coordinates is not implemented at the hemisphere boundaries such as at the equator (0 deg. N.), or at 0 deg. longitude. The input data must be completely above or below the equator and similarly completely east or west of 0 deg. longitude.

C. PERFORMANCE

Forming the 3-D perspective views requires the input data to be transformed from one 3-D coordinate system to

the image plane system and ultimately to the screen coordinate system. As discussed in Chapter Three, the size of the input data files, specifically the elevation data will affect the performance of the program. Table 6-1 is a breakdown of the amount of time required to complete each of the major sections of the program. The overall performance is satisfactory and breaks down to 40 seconds to complete a view after the observer location is entered. The plotting of the view on the screen is 89% of this time, while the calculation involved accounts for only 11% of the time. Clearly the plotting time must be improved prior to spending an appreciable amount of time in the calculation subroutines. The length of time required by the plotting routine is related to the number of triangles being passed to the graphics hardware. The communication involved in passing the vertices of each panel to the graphics processor is the bottleneck. Improvements may be achieved in this area by writing a driver to directly load the appropriate values to the graphics hardware and bypassing the overhead of the software routines presently utilized.

The combination of the perspective transformation and the hidden surface removal routines account for 95% of the 11% calculation time as seen in Table 6-1. Other methods of achieving slight performance increases may include a hardware implementation of the perspective transformation

TABLE 6-1 COMPUTATION TIME

Subsection	Resolution 61 x 61 (time in seconds)	13 x 13
Set up graphics	.80	.53
User input	---	---
Read elev. file	.25	.11
Read image file	10.22	10.69
Form triangles	3.3	.3
Average gray shade	6.87	2.54
Observer location	---	---
Form [M] matrix	.01	.01
[M] multiply	2.51	.12
Affine transform	.23	.01
Hidden surface removal	1.79	.07
Plotting	35.24	1.76
Total elapsed after Observer location	39.78	1.97
Calculation time percentage	11.4%	10.6%

matrix multiplies and a more efficient hidden surface removal routine. Altering the hidden surface removal routine most likely will result in more calculation time required in that area, but may result in substantial savings in the plotting time. The approach for hidden

surface removal used here does not actually remove hidden triangles but draws them in order from the background to foreground. By altering the hidden surface routine so that hidden panels are not plotted at all, a savings may be made in the time required to plot the view.

D. FUTURE WORK

This program is a first step in the direction of combining various layers of data into a single form. There are several areas which are being considered for further work.

The type of data merged onto the elevation grid is not limited to the imagery data used here. Magnetic anomaly data is available and could also be merged onto the grid. This would permit the observer to make correlations between the terrain and the magnetic response.

The original effort in this area began with aerial photographs and surface terrain. Applying the terrain elevation data and an aerial photograph to this program would broaden the versatility of the program greatly.

Combining two layers of data is a starting point, the ability to combine three or four layers onto the elevation grid using colors to distinguish the layers may be possible.

In order to make this program truly useful, a user interface must be designed. As a minimum this requires a

method of entering observer locations based on the current location and perspective view. A particularly effective user interface would involve using a pointer to select an area on the screen and tell the program "I wish to go there". This would permit "driving around" an area of interest.

Finally, a method of loading data from the storage device and merging it with the currently loaded data is required. This would remove the boundaries from the image displayed on the screen so that when the observer location is near the edge of the current area, the adjacent areas will be loaded and displayed.

APPENDIX A. PROGRAM SUMMARY

This appendix lists the different routines of the program by subroutine name and briefly lists the inputs and outputs of the subroutine. Also given are the names of the subroutines which interact with each subroutine.

1. SONAR3D

   A. Function

      Main routine, calls other supporting routines to complete the 3-D perspective view

   B. Input

      None

   C. Output

      None

   D. Calling routine

      None

   E. Called routine

      INPUT

      READ_ELEV

      READ_IMAGE

      TER_XYZ

      IM_REFAVG

      OBS_LOC

      M_ORIEN

NEW_IJ

XY2IJ

HIDDEN_SURF

CLRSCRN

2. Subroutine INPUT

A. Function

Reads in or creates the initial data required for the elevation data file and the image data file.

B. Input

NAME - name of input data file

ELFILE - elevation data file name

EL_SIZ - elevation data number of columns

EL_REC - elevation data number of rows

IMFILE - image data file name

IM_SIZ - image data number of columns

IM_REC - image data number of rows

ILAD,ILAM,ILAS - northwest latitude of image file

ILOD,ILOM,ILOS - northwest longitude of image file

FLAD,FLAM,FLAS - southeast latitude of image file

FLAD,FLAM,FLAS - southeast longitude of image file

TITLE - 50 character title placed on the image screen

C. Output

Same as input

D. Calling routines

SONAR3D

E. Called routines

CLRSCRN

3. Subroutine READ_ELEV

A. Function

Read in the elevation data file from disk

B. Input

None

C. Output

ILATD,ILATM,ILATS - southwest latitude of the elevation data file

ILOND,ILONM,ILONS - southwest longitude of the elevation data file

D_LAT - latitude grid size in seconds

D_LONG - longitude grid size in seconds

IENDM - number of rows of the elevation data

IENDN - number of columns of the elevation data

RELEV(*,*) - array containing the elevation data points

D. Calling Routines

SONAR3D

E. Called routines

None

4. Subroutine READ_IMAGE

   A. Function

      Read in the image data file from disk

   B. Input

      IM_SIZ - number of columns of the image data

      IM_REC - number of rows of the image data

   C. Output

      IMAGE(*,*) - array containing the image data

   D. Calling routines

      SONAR3D

   E. Called routines

      None

5. Subroutine TER_XYZ

   A. Function

      Converts each elevation point to it's geo-rectangular XYZ coordinates and calculates the corresponding image row/column coordinates for each point

   B. Input

      LATD,LATM,LATS - reference elevation latitude

      ILOND,LONM,LONS - reference elevation longitude

      D_LAT - latitude grid size in seconds

      D_LONG - longitude grid size in seconds

      RELEV(*,*) - array holding the elevation data

      IENDM - number of rows of the elevation data

IENDN - number of columns of the elevation data

ILAD,ILAM,ILAS - reference northwest image latitude

ILOD,ILOM,ILOS - reference northwest image longitude

FLAD,FLAM,FLAS - reference southeast image latitude

FLOD,FLOM,FLOS - reference southeast image longitude

IM_SIZ - number of columns of the image data

IM_REC - number of rows of the image data

C. Output

IA(*) - array containing the image column coordinates

JA(*) - array containing the image row coordinates

XYZ(*,3) - array holding the (X,Y,Z) coordinates of each elevation point

D. Calling routines

SONAR3D

E. Called routines

CONV2SEC

DMS2XYZ

6. Subroutine IM_REFAVG

A. Function

This routine forms the panel array containing the nodes of the triangles and the average gray shade.

B. Inputs

IA(*) - array containing the image column coordinates

JA(*) - array containing the image row coordinates

IENDM - number of rows of the elevation data

IENDN - number of columns of the elevation data

IMAGE(*,*) - array containing the image data

C. Outputs

PL_AR(*,5) - array containing the nodes of the triangle and the gray shade

D. Calling routines

SONAR3D

E. Called routines

None

7. Subroutine OBS_LOC

A. Function

Calculates the observer location in (X,Y,Z) coordinates given the latitude and longitude.

B. Input

LATD,LATM,LATS - Latitude of observer location

LOND,LONM,LONS - Longitude of observer location

HEIGHT - Observer elevation in meters

CTS - Observer course in degrees

C. Outputs

OBS_LOC(*) - array holding observer location parameters

X1,Y1,Z1 - Geo-rectangular coordinates of observer location

X2,Y2,Z2 - Geo-rectangular coordinates of second observer point along LOS

D. Calling routines

SONAR3D

E. Called routines

DMS2XYZ

8. Subroutine M_ORIEN

A. Function

Determine the orientation of the image plane and calculate the [M] matrix parameters

B. Inputs

X1,Y1,Z1 - Geo-rectangular coordinates of observer location

X2,Y2,Z2 - Geo-rectangular coordinates of second observer point along LOS

C. Outputs

M(3,3) - [M] matrix parameters

D. Calling routines

SONAR3D

E. Called routines

None

9. Subroutine NEW_IJ

A. Function

Calculate the image plane coordinates of all object points which are in front of the field of view. Also mark as HIDDEN any nodes behind the image plane

B. Inputs

X1,Y1,Z1 - Geo-rectangular coordinates of the observer location

X2,Y2,Z2 - Geo-rectangular coordinates of the second observer point along LOS

ITOT - Total number of elevation points

XYZ(*,3) - array holding the (X,Y,Z) coordinates of each elevation point

FOCUS - Focal length

M(3,3) - [M] matrix parameters

C. Outputs

IMAX(*) - x image coordinate

IMAY(*) - y image coordinate

DEPTH(*) - Distance from observer of each elevation point

HIDDEN(*) - Flag for points hidden behind image plane

D. Calling routines

SONAR3D

E. Called routines

None

10. Subroutine XY2IJ

A. Function

Convert the image (x,y) coordinates to screen (i,j) coordinates

B. Inputs

IMAX(*) - x image coordinate

IMAY(*) - y image coordinate

ITOT - Total number of elevation points

XIMA_MAX - image frame size (x direction)

YIMA_MAX - image frame size (y direction)

HIDDEN(*) - Flag for points hidden behind image plane

C. Outputs

IA(*) - screen x coordinate

JA(*) - screen y coordinate

D. Calling routines

SONAR3D

E. Called routines

AFFIN

11. Subroutine AFFIN

A. Function

Calculates the coefficients for the AFFIN transform from image coordinates to screen coordinates

B. Inputs

XIMA_MAX - image frame size (x direction)

YIMA_MAX - image frame size (y direction)

C. Outputs

A1,A2,B1,B2,C1,C2 - parameters for the AFFIN transform

D. Calling routines

XY2IJ

E. Called routines

None

12. Subroutine HIDDEN_SURF

A. Function

Remove hidden surfaces and plot to screen

B. Inputs

IA(*) - screen x coordinate

JA(*) - screen y coordinate

IENDM - Number of rows of the elevation data

IENDN - Number of columns of the elevation data

DEPTH(*) - Distance from the observer of each elevation point

PL_AR(*,5) - array containing the nodes of the triangle and the gray shade

HIDDEN(*) - Flag for points hidden behind the image plane

VD_ID - Virtual display identifier

WD_ID - Window identifier

C. Output

None

D. Calling routine

SONAR3D

E. Called routine

QUICKSORT

UISDC$ERASE

UIS$SET_WRITING_INDEX

UISDC$PLOT

13. Subroutine QUICKSORT

A. Function

Sort the panels based on maximum distance from the observer

B. Input

ARRAY(*,5) - array to sort

KEY(*) - index to main array

COUNT - number of elements to sort

C. Output

ARRAY(*,5) - original array

KEY(*) - index to main array (revised order)

D. Calling routine

HIDDEN_SURF

E. Called routine

None

14. Subroutine CLRSCRN

A. Function

Clear the screen

B. Input

None

C. Output

None

D. Calling routine

SONAR3D

INPUT

E. Called routines

None

15. Subroutine CONV2SEC

A. Function

Convert DEG,MIN,SEC coordinates to seconds

B. Input

DEG,MIN,SEC - Latitude or longitude in deg,min,sec format

C. Output

SEC - Total number of seconds

D. Calling routine

TER_XYZ

E. Called routine

None

16. Subroutine DMS2XYZ

A. Function

Convert DMS data to (X,Y,Z) coordinates

B. Input

LATD,LATM,LATS - Latitude in deg,min,sec format

LOND,LONM,LONS - Longitude in deg,min,sec format

HEIGHT - elevation with respect to sea level in meters

C. Output

X,Y,Z - (X,Y,Z) coordinates of input point

D. Calling routine

TER_XYZ

OBS_LOC

E. Called routine

None

APPENDIX B. PROGRAM LISTING

```
      PROGRAM SONAR3D
C ************************************************************
C
C       THIS PROGRAM CONSTRUCTS THE ELEVATION FILE AND IMAGE
C       FILE THAT IS REQUIRED BY THE 3-D PROGRAM. MAXIMUM
C       ELEVATION ARRAY SIZE IS 70 ROWS x 70 COLUMNS. MAXIMUM
C       IMAGE SIZE IS 1024 x 1024.
C
C ************************************************************
      IMPLICIT INTEGER (A-Z)
C
C get graphics libraries
C
      INCLUDE 'SYS$LIBRARY:UISUSRDEF'
      INCLUDE 'SYS$LIBRARY:UISENTRY'
      INCLUDE 'SYS$LIBRARY:UISMSG'
C
C  the following variables define the elevation file maximum size
C  and the image file maximum size.  In order to simplify altering
C  of these values, they have been grouped together.  Change the
C  values in the DATA statement, and in the definition statements
C  which immediately follow.
C
      COMMON /ELEV/ROW_ELEV,COL_ELEV,TOT_ELEV,NPLANES
      COMMON /IMAGE/ROW_IMAGE,COL_IMAGE
      DATA ROW_ELEV,COL_ELEV,TOT_ELEV,NPLANES/70,70,4900,10000/
      DATA ROW_IMAGE,COL_IMAGE/1024,1024/
C
      BYTE IMAGE(1024,1024)
      INTEGER HIDDEN(4900),IA(4900),JA(4900)
      INTEGER PL_AR(10000,5),DEPTH_KEY(10000)
      REAL RELEV(70,70)
      REAL*8 DEPTH(4900),XYZ(4900,3),IMAX(4900),IMAY(4900)
C
      CHARACTER ELFILE*13,IMFILE*13,FILE_NAME*13,ANS*1,TITLE*50
C
      INTEGER IENDN,IENDM,ITOT,ILATD,ILATM,ILATS
      INTEGER ILOND,ILONM,ILONS
      INTEGER D_LAT,D_LONG,LATS,LONS,OBSLOC(8)
      INTEGER EL_SIZ,EL_REC,IM_SIZ,IM_REC
      INTEGER ILAD,ILAM,ILAS,ILOD,ILOM,ILOS
      INTEGER FLAD,FLAM,FLAS,FLOD,FLOM,FLOS
C
      REAL*4 I_VECTOR
      REAL*8 X1,Y1,Z1,X2,Y2,Z2,M(3,3)
```

```
      REAL*8 FOCUS,XFRAME,YFRAME
      REAL*8 HEIGHT,PWR,SCALE
C
C set up graphics environment
C
C create a color map with 251 entries
C
      DATA VCM_SIZE/251/
      VCM_ID=UIS$CREATE_COLOR_MAP(VCM_SIZE)
C
C make a virtual display 16.8 cm x 16.8 cm
C
      VD_ID=UIS$CREATE_DISPLAY(0.0,0.0,512.0,512.0,16.8,16.8,VCM_ID)
C
C fill in the color map
C
        DO 50 I=0,250
          I_VECTOR=I/250.
          CALL UIS$SET_INTENSITY(VD_ID,I,I_VECTOR)
50      CONTINUE
C
C set the background color and fill pattern
C
      CALL UIS$SET_INTENSITY(VD_ID,0,0.0)
      CALL UIS$SET_FONT(VD_ID,1,1,'UIS$FILL_PATTERNS')
      CALL UIS$SET_FILL_PATTERN(VD_ID,1,1,PATT$C_FOREGROUND)
C
C start main routine, read input
C
      CALL INPUT(ELFILE,EL_SIZ,EL_REC,
     .   IMFILE,IM_SIZ,IM_REC,ILAD,ILAM,ILAS,ILOD,ILOM,ILOS,
     .   FLAD,FLAM,FLAS,FLOD,FLOM,FLOS,TITLE)
C
      EL_REC=EL_REC+1
      IM_SIZ=IM_SIZ/4
C
C open the input data files
C
      OPEN(UNIT=1,FILE=ELFILE,STATUS='OLD',ACCESS='DIRECT',
     .     RECORDSIZE=EL_SIZ,MAXREC=EL_REC)
      OPEN(UNIT=4,FILE=IMFILE,STATUS='OLD',ACCESS='DIRECT',
     .     RECORDSIZE=IM_SIZ,MAXREC=IM_REC)
C
      CALL READ_ELEV(ILATD,ILATM,ILATS,ILOND,ILONM,ILONS,
     .     D_LAT,D_LONG,RELEV,IENDN,IENDM)
      CALL READ_IMAGE(IMAGE,IM_SIZ,IM_REC)
      CALL TER_XYZ(ILATD,ILATM,ILATS,ILOND,ILONM,ILONS,
     .     D_LAT,D_LONG,RELEV,IENDM,IENDN,XYZ,IA,JA,
     .     ILAD,ILAM,ILAS,ILOD,ILOM,ILOS,
     .     FLAD,FLAM,FLAS,FLOD,FLOM,FLOS,IM_SIZ,IM_REC)
C
      CALL IM_REFAVG(IA,JA,IMAGE,IENDM,IENDN,PL_AR)
```

```
      YFRAME=29000.0
      XFRAME=35000.0
      FILE_NAME=' '
      SAVE_FLAG=0
      FOCUS=.050
      PWR=1.0
      SCALE=1.2
      ITOT=IENDN*IENDM
      WD_ID=UIS$CREATE_WINDOW(VD_ID,'SYS$WORKSTATION',TITLE,
     .   0.0,0.0,512.0,512.0,16.8,16.8)
C
C
60    CALL OBS_LOC(X1,Y1,Z1,X2,Y2,Z2,OBSLOC)
      CALL M_ORIEN(M,X1,Y1,Z1,X2,Y2,Z2)
      CALL NEW_IJ(X1,Y1,Z1,X2,Y2,Z2,ITOT,XYZ,FOCUS,M,IMAX,
     .     IMAY,HIDDEN,DEPTH)
70    CALL XY2IJ(IA,JA,IMAX,IMAY,XFRAME,YFRAME,ITOT,HIDDEN)
      CALL HIDDEN_SURF(IA,JA,IENDM,IENDN,DEPTH,PL_AR,VD_ID,
     .  HIDDEN,DEPTH_KEY,WD_ID)
C
C type the observer location and menu
C
80    CALL CLRSCRN
      WRITE(6,*)'OBSERVER LOCATION'
      WRITE(6,*)
      WRITE(6,85)'LAT: ',OBSLOC(1),OBSLOC(2),OBSLOC(3)
      WRITE(6,85)'LONG: ',OBSLOC(4),OBSLOC(5),OBSLOC(6)
      WRITE(6,90)'HEIGHT: ',OBSLOC(7),'  HEADING: ',OBSLOC(8)
      WRITE(6,91)'MAGNIFICATION: ',PWR,'   YSCALE: ',SCALE
85    FORMAT(A8,I4,I3,I3)
90    FORMAT(A9,I7,A12,I3)
91    FORMAT(A16,F4.1,A12,F4.1)
      WRITE(6,*)
      WRITE(6,*)'SELECT ONE OF THE FOLLOWING'
        WRITE(6,*)'     (M)agnification'
        WRITE(6,*)'     (Y)scale factor'
        WRITE(6,*)'     (O)bserver Location'
        WRITE(6,*)'     (S)ave Image'
        WRITE(6,*)'     (E)xit'
C
        READ(5,100) ANS
100     FORMAT(A1)
C
        IF (ANS .NE. 'M' .AND. ANS .NE. 'm') GO TO 105
          WRITE(6,*)'ENTER MAGNIFICATION FACTOR'
          READ(5,*)PWR
102       XFRAME=35000.0/PWR
          YFRAME=35000.0/(PWR*SCALE)
          GO TO 70
C
105     IF (ANS .NE. 'Y'.AND. ANS .NE. 'y') GO TO 110
          WRITE(6,*)'ENTER Y-SCALE FACTOR'
```

```
          READ(5,*)SCALE
          GO TO 102
C
110     IF (ANS .NE. 'O'.AND. ANS .NE. 'o') GO TO 115
          GO TO 60
C
115     IF (ANS .NE. 'S' .AND. ANS .NE. 's') GO TO 118
          IF (FILE_NAME .EQ. ' ') THEN
            WRITE(6,*)'ENTER THE NAME OF THE IMAGE FILE'
            READ(5,116) FILE_NAME
          ENDIF
          SAVE_FLAG=SAVE_FLAG+1
          CALL IMAGE_SAVE(WD_ID,FILE_NAME,SAVE_FLAG)
          GO TO 80
116     FORMAT(A13)
118     IF (ANS .NE. 'E'.AND. ANS .NE. 'e') GO TO 80
C
120     IF (SAVE_FLAG .GT. 0) THEN
          WRITE(11) SAVE_FLAG
        ENDIF
        CLOSE(1)
        CLOSE(4)
        CLOSE(10,STATUS='SAVE')
        CLOSE(11,STATUS='SAVE')
        CALL UIS$DELETE_DISPLAY(VD_ID)
        CALL CLRSCRN
        END
```

```
C
C ********************************************************
C
      SUBROUTINE INPUT(ELFILE,EL_SIZ,EL_REC,IMFILE,
     .   IM_SIZ,IM_REC,ILAD,ILAM,ILAS,ILOD,ILOM,ILOS,
     .   FLAD,FLAM,FLAS,FLOD,FLOM,FLOS,TITLE)
C
C      INPUTS = ELFILE   elevation data file name
C               EL_SIZ   # column of ELFILE
C               EL_REC   # rows of ELFILE
C               IMFILE   image data file name
C               IM_SIZ   # columns of IMFILE
C               IM_REC   # rows of IMFILE
C               ILAD,ILAM,ILAS
C               ILOD,ILOM,ILOS  Northwest corner of image file
C               FLAD,FLAM,FLAS
C               FLOD,FLOM,FLOS  Southeast corner of image file
C               NAME     file name for the input data file
C               TITLE    header name for the image
C
C     OUTPUTS = NONE
C
C ********************************************************
C
      IMPLICIT INTEGER (A-Z)
C
      CHARACTER ELFILE*13,IMFILE*13,TITLE*50,ANS*1,NAME*13
C
      INTEGER EL_SIZ,EL_REC,IM_SIZ,IM_REC
      INTEGER ILAD,ILAM,ILAS,FLAD,FLAM,FLAS
      INTEGER ILOD,ILOM,ILOS,FLOD,FLOM,FLOS
C
C start routine here
C
5     CALL CLRSCRN
      WRITE(6,*)'Do you wish to create an input data file (Y/N)? '
      READ(5,100) ANS
100   FORMAT(A1)
      IF (ANS .EQ. 'Y' .OR. ANS .EQ. 'y') GO TO 200
      IF (ANS .EQ. 'N' .OR. ANS .EQ. 'n') GO TO 500
      GO TO 5
C
C create input file
C
200   CALL CLRSCRN
      WRITE(6,*)'Input the filename for the input file(*.dat): '
      READ(5,205) NAME
      OPEN (UNIT=1,NAME=NAME,STATUS='NEW',
     .   ACCESS='SEQUENTIAL',FORM='FORMATTED')
C
C get elevation file data
C
```

```
        WRITE(6,*)'Enter the elevation data file name: '
        READ(5,205) ELFILE
        WRITE(6,*)'Elevation file number of columns: '
        READ(5,*) EL_SIZ
        WRITE(6,*)'Elevation file number of rows: '
        READ(5,*) EL_REC
C
C get image file data
C
        WRITE(6,*)'Enter the image data file name: '
        READ(5,205) IMFILE
        WRITE(6,*)'Image file number of columns: '
        READ(5,*) IM_SIZ
        WRITE(6,*)'Image file number of rows: '
        READ(5,*) IM_REC
C
C northwest corner of image latitude and longitude
C
        WRITE(6,*)'Enter the North-West lat/long of image'
        WRITE(6,*)'Input the latitude (DEG,MIN,SEC): '
        READ(5,*)ILAD,ILAM,ILAS
        WRITE(6,*)'Input the longitude (DEG,MIN,SEC): '
        READ(5,*)ILOD,ILOM,ILOS
C
C southeast corner of image latitude and longitude
C
        WRITE(6,*)'Enter the South-East lat/long of image'
        WRITE(6,*)'Input the latitude (DEG,MIN,SEC): '
        READ(5,*) FLAD,FLAM,FLAS
        WRITE(6,*)'Input the longitude (DEG,MIN,SEC): '
        READ(5,*) FLOD,FLOM,FLOS
        WRITE(6,*)'Input a title for the image (50 char max.): '
        READ(5,206) TITLE
C
C write data to data file
C
        WRITE(1,205) ELFILE
        WRITE(1,207) EL_SIZ
        WRITE(1,207) EL_REC
        WRITE(1,205) IMFILE
        WRITE(1,207) IM_SIZ
        WRITE(1,207) IM_REC
        WRITE(1,209) ILAD,ILAM,ILAS
        WRITE(1,210) ILOD,ILOM,ILOS
        WRITE(1,209) FLAD,FLAM,FLAS
        WRITE(1,210) FLOD,FLOM,FLOS
        WRITE(1,206) TITLE
C
205     FORMAT(A13)
206     FORMAT(A50)
207     FORMAT(I3)
209     FORMAT(3I3)
```

```
210     FORMAT(I4,2I3)
        GO TO 600
C
C read an input file
C
500     CALL CLRSCRN
        WRITE(6,*)'Enter the name of the input data file: '
        READ(5,205) NAME
        OPEN(UNIT=1,NAME=NAME,STATUS='OLD',
     .    ACCESS='SEQUENTIAL',FORM='FORMATTED')
        READ(1,205) ELFILE
        READ(1,207) EL_SIZ
        READ(1,207) EL_REC
        READ(1,205) IMFILE
        READ(1,207) IM_SIZ
        READ(1,207) IM_REC
        READ(1,209) ILAD,ILAM,ILAS
        READ(1,210) ILOD,ILOM,ILOS
        READ(1,209) FLAD,FLAM,FLAS
        READ(1,210) FLOD,FLOM,FLOS
        READ(1,206) TITLE
C
600     CLOSE (UNIT=1)
        RETURN
        END
```

```
C
C ***************************************************************
C
      SUBROUTINE READ_ELEV(ILATD,ILATM,ILATS,ILOND,ILONM,ILONS,
     .    D_LAT,D_LONG,RELEV,IENDN,IENDM)
C
C     THIS SUBROUTINE READS THE ELEVATION FILE AND PLACES THE DATA
C     INTO A REAL 50X50 ARRAY  RELEV().
C     INPUTS = NONE
C
C      OUTPUT = ILATD   southwest latitude of elev data (degrees)
C               ILATM   southwest latitude of elev data (minutes)
C               ILATS   southwest latitude of elev data (seconds)
C               ILOND   southwest longitude of elev data (degrees)
C               ILONM   southwest longitude of elev data (minutes)
C               ILONS   southwest longitude of elev data (seconds)
C               D_LAT   grid size in minutes of latitude
C               D_LONG  grid size in minutes of longitude
C               RELEV() elev data
C               IENDM   # rows of elev data
C               IENDN   # columns of elev data
C
C ***************************************************************
      IMPLICIT INTEGER (A-Z)
C
      COMMON /ELEV/ROW_ELEV,COL_ELEV
      INTEGER IENDN,IENDM,D_LAT,D_LONG
      INTEGER ILATD,ILATM,ILATS,ILOND,ILONM,ILONS
C
      REAL RELEV(ROW_ELEV,COL_ELEV)
C
      READ(1,REC=1)ILATD,ILATM,ILATS,ILOND,ILONM,ILONS,
     .    D_LAT,D_LONG,IENDN,IENDM
C
      DO 10 M=1,IENDM
        READ(1,REC=M+1)(RELEV(M,N),N=1,IENDN)
10    CONTINUE
      RETURN
      END
```

```
C
C ************************************************************
C
        SUBROUTINE READ_IMAGE(IMAGE,IM_SIZ,IM_REC)
C
C       THIS SUBROUTINE READS THE IMAGE DATA INTO AN ARRAY.
C         INPUTS = IM_SIZ   number of columns of image
C                  IM_REC   number of rows of image
C
C         OUTPUT = IMAGE()  image gray level file
C
C************************************************************
      IMPLICIT INTEGER (A-Z)
      COMMON /IMAGE/ROW_IMAGE,COL_IMAGE
C
      BYTE IMAGE(ROW_IMAGE,COL_IMAGE)
C
      INTEGER IM_SIZ,IM_REC
C
      DO 10 IR=1,IM_REC
       READ(4,REC=IR)(IMAGE(IR,IC),IC=1,IM_SIZ*4)
10    CONTINUE
      RETURN
      END
```

```
C
C ****************************************************************
C
      SUBROUTINE TER_XYZ(LATD,LATM,LATS,ILOND,LONM,LONS,
     . D_LAT,D_LONG,RELEV,IENDM,IENDN,XYZ,IA,JA,
     . ILAD,ILAM,ILAS,ILOD,ILOM,ILOS,
     . FLAD,FLAM,FLAS,FLOD,FLOM,FLOS,IM_SIZ,IM_REC)
C
C     THIS SUBROUTINE CONVERTS EACH ELEVATION POINT TO ITS DMS
C     EQUIVALENT. IT USES THE FACT THAT EACH POINT REPRESENTS
C     AN EQUAL CHANGE FROM THE LAST. THE REFERENCE LAT/LONG
C     AND THE DELTA LAT/LONG ARE PASSED IN THE SUBROUTINE CALL.
C
C     ...ASSUMES COLOR COMPLETELY OVERLAPS THE ELEV DATA
C     ...NEED TO KNOW THE LAT/LONG OF THE COLOR IMAGE
C     ...ALSO CALCS THE IA(), JA() COORDINATES FOR EACH ELEV PT
C
C                   ELEVATION DATA
C     INPUTS     LATD - REFERENCE LATITUDE (DEGREES)
C                LATM - REFERENCE LATITUDE (MINUTES)
C                LATS - REFERENCE LATITUDE (SECONDS)
C                ILOND - REFERENCE LONGITUDE (DEGREES)
C                LONM - REFERENCE LONGITUDE (MINUTES)
C                LONS - REFERENCE LONGITUDE (SECONDS)
C                D_LAT - DISTANCE BETWEEN ROWS
C                D_LONG - DISTANCE BETWEEN COLUMNS
C                RELEV() - ELEVATION DATA FILE
C                IENDM - NUMBER OF ROWS
C                IENDN - NUMBER OF COLUMNS
C
C     OUTPUTS    XYZ   - OUTPUT DATA FILE
C
C ****************************************************************
      IMPLICIT INTEGER (A-Z)
      COMMON /ELEV/ ROW_ELEV,COL_ELEV,TOT_ELEV
C
      INTEGER IA(TOT_ELEV),JA(TOT_ELEV)
      INTEGER LATD,LATM,LATS,ILOND,ILONM,D_LAT,D_LONG
      INTEGER LOND,LONM,LONS
      INTEGER ILAD,ILAM,ILAS,ILOD,ILOM,ILOS
      INTEGER FLAD,FLAM,FLAS,FLOD,FLOM,FLOS
      INTEGER IM_SIZ,IM_REC
C
      REAL RELEV(ROW_ELEV,COL_ELEV)
      REAL*8 HEIGHT,X,Y,Z,XYZ(TOT_ELEV,3)
C
C convert to seconds
C
       CALL CONV2SEC(ILAD,ILAM,ILAS)
       CALL CONV2SEC(ILOD,ILOM,ILOS)
       CALL CONV2SEC(FLAD,FLAM,FLAS)
       CALL CONV2SEC(FLOD,FLOM,FLOS)
```

```
C
        DELLAT=ILAS-FLAS
        DELLON=FLOS-ILOS
C
C set flag for proper hemisphere
C
        IF (ILAD .GE. 0) THEN
          LAT_FLAG=1
        ELSE
          LAT_FLAG=-1
        ENDIF
        IF (ILOD .GE. 0) THEN
          LON_FLAG=1
        ELSE
          LON_FLAG=-1
        ENDIF
C
C find initial seconds of elev. data
C
      CALL CONV2SEC(LATD,LATM,LATS)
      CALL CONV2SEC(ILOND,LONM,LONS)
C
C set up for 1st increment
C
      LATS=LATS-D_LAT
      ILONS=LONS-D_LONG
C
C start routine here
C
        IR=0
        DO 10 M=1,IENDM
         LATS=LATS+D_LAT
         LONS=ILONS
         DO 20 N=1,IENDN
           IR=IR+1
           LONS=LONS+D_LONG
           HEIGHT=RELEV(M,N)
           ROW=INT((IM_REC-1)*(ILAS-LATS)/DELLAT)+1
           COL=INT(((IM_SIZ*4)-1)*(LONS-ILOS)/DELLON)+1
C
C the row and column coordinates are 1-512
C
           IA(IR)=COL
           JA(IR)=ROW
           CALL DMS2XYZ (0,0,LATS,0,0,LONS,HEIGHT,X,Y,Z)
           XYZ(IR,1)=X
           XYZ(IR,2)=Y
           XYZ(IR,3)=Z
20       CONTINUE
10     CONTINUE
       RETURN
       END
```

```
C
C ***************************************************************
C
      SUBROUTINE DMS2XYZ(LATD,LATM,LATS,LOND,LONM,LONS,HEIGHT,X,Y,Z)
C
C     THIS SUBROUTINE CONVERTS DMS DATA TO X,Y,Z.
C
C      INPUTS = LATD    latitude in degrees
C               LATM    latitude in minutes
C               LATS    latitude in seconds
C               LOND    longitude in degrees
C               LONM    longitude in minutes
C               LONS    longitude in seconds
C               HEIGHT  height above sea level (meters)
C
C      OUTPUT = X,Y,Z   XYZ coordinates of DMS point
C
C ***************************************************************
      IMPLICIT INTEGER (A-Z)
C
      INTEGER LATD,LATM,LATS,LOND,LONM,LONS
C
      REAL*8 PHI,LAMDA,N,X,Y,Z,HEIGHT
      REAL*8 PI,C1,C2,C3,E_SQUARE,A,RADIAN
C
      PARAMETER(PI=3.14159265358793238)
      PARAMETER(C1=180.,C2=60.,C3=3600.)
      PARAMETER(E_SQUARE=0.006768658,A=6378206.4)
C
      RADIAN=PI/C1
C
      IF (LATD .GE. 0) THEN
       PHI=(LATD+LATM/C2+LATS/C3)*RADIAN
      ELSE
       PHI=(LATD-LATM/C2-LATS/C3)*RADIAN
      END IF
C
      IF (LOND .GE. 0) THEN
       LAMDA=(LOND+LONM/C2+LONS/C3)*RADIAN
      ELSE
       LAMDA=(LOND-LONM/C2-LONS/C3)*RADIAN
      END IF
C
      N=A/SQRT(1-E_SQUARE*SIN(PHI)*SIN(PHI))
      X=(N+HEIGHT)*COS(PHI)*COS(LAMDA)
      Y=(N+HEIGHT)*COS(PHI)*SIN(LAMDA)
      Z=(N*(1-E_SQUARE)+HEIGHT)*SIN(PHI)
      RETURN
      END
```

```
C
C*************************************************************
C
      SUBROUTINE IM_REFAVG(IA,JA,IMAGE,IENDM,IENDN,PL_AR)
C
C     THIS SUBROUTINE CONSTRUCTS THE GEOMETRY FILE
C     THAT CONTAINS THE THREE NODAL POINTS THAT MAKE UP
C     AN IMAGE PLANE AND THE GREY SCALE VALUE ASSIGNED TO
C     THAT PLANE.
C       INPUTS = IMAGE()    image gray level file
C                IENDM      # rows of elev data
C                IENDN      # columns of elev data
C                IA()       image column index
C                JA()       image row index
C
C       OUTPUT=  PL_AR()    array holding nodes and
C                           gray value for each panel
C
C ************************************************************
C
      IMPLICIT INTEGER (A-Z)
C
      COMMON /ELEV/ROW_ELEV,COL_ELEV,TOT_ELEV,NPLANES
      COMMON /IMAGE/ROW_IMAGE,COL_IMAGE
C
      BYTE IMAGE(ROW_IMAGE,COL_IMAGE)
C
      INTEGER IA(TOT_ELEV),JA(TOT_ELEV),PL_AR(NPLANES,5)
      INTEGER IY,M,N,IGREY1,IGREY2,L,L1
      INTEGER IENDM,IENDN,NODE_A,NODE_B,NODE_C,NODE_D,IR
      INTEGER ICOUNT1,ICOUNT2,ITOT1,ITOT2
C
      REAL*8 SLOPE,YINT
C
      DO 90 IR=1,IENDM-1
       IL=(IR-1)*IENDN
       DO 80 N=1+IL,IENDN-1+IL
        NODE_A=N
        NODE_B=N+1
        NODE_C=N+IENDN
        NODE_D=NODE_C+1
        SLOPE=(JA(NODE_C)-JA(NODE_B))*1.0/(IA(NODE_C)-IA(NODE_B))*1.0
        YINT=1.0*JA(NODE_B)-SLOPE*IA(NODE_B)
C
C now know the slope and y-intecept of the line forming the triangle
C
        ITOT1=0
        ITOT2=0
        ICOUNT1=0
        ICOUNT2=0
        DO 70 M=IA(NODE_B),IA(NODE_A),-1
         IY=SLOPE*M+YINT
```

```
        DO 50 L=JA(NODE_B),IY,-1
         IF (IMAGE(L,M) .LT. 0) THEN
          ITOT1=ITOT1+IMAGE(L,M)+256
         ELSE
         ITOT1=ITOT1+IMAGE(L,M)
         END IF
         ICOUNT1=ICOUNT1+1
50      CONTINUE
        IF(M.LT.IA(NODE_B))THEN
         DO 60 L=IY,JA(NODE_C),-1
          IF (IMAGE(L,M) .LT. 0) THEN
           ITOT2=ITOT2+IMAGE(L,M)+256
          ELSE
           ITOT2=ITOT2+IMAGE(L,M)
          END IF
          ICOUNT2=ICOUNT2+1
60       CONTINUE
        END IF
70     CONTINUE
       L1=(N-(IR-1))*2
       IGREY1=ITOT1/ICOUNT1
       IGREY2=ITOT2/ICOUNT2
       PL_AR(L1-1,1)=NODE_A
       PL_AR(L1-1,2)=NODE_B
       PL_AR(L1-1,3)=NODE_C
       PL_AR(L1-1,4)=IGREY1
       PL_AR(L1,1)=NODE_B
       PL_AR(L1,2)=NODE_D
       PL_AR(L1,3)=NODE_C
       PL_AR(L1,4)=IGREY2
80    CONTINUE
90   CONTINUE
     RETURN
     END
```

```
C
C ************************************************************
C
      SUBROUTINE OBS_LOC(X1,Y1,Z1,X2,Y2,Z2,OBSLOC)
C
C     THIS SUBROUTINE CALCULATES THE NEW OBSERVER XYZ
C     LOCATION FROM DESIRED LAT. AND LONG. INPUTS.
C
C     INPUTS  =  NONE
C     OUTPUTS =  X1,Y1,Z1       observer location
C                X2,Y2,Z2       observer location direction
C                OBSLOC()      observer location array
C
C ************************************************************
      IMPLICIT INTEGER (A-Z)
C
      CHARACTER ANS*1
C
      INTEGER LATD,LATM,LATS,LOND,LONM,LONS,CTS
      INTEGER OBSLOC(8)
C
      REAL*8 X1,Y1,Z1,X2,Y2,Z2,HEIGHT
C
C read in last values of observer location
C
      LATD=OBSLOC(1)
      LATM=OBSLOC(2)
      LATS=OBSLOC(3)
      LOND=OBSLOC(4)
      LONM=OBSLOC(5)
      LONS=OBSLOC(6)
      HEIGHT=OBSLOC(7)
      CTS=OBSLOC(8)
C
      FLAG=0
1     CALL CLRSCRN
      WRITE(6,*)'OBSERVER LOCATION'
      WRITE(6,*)
      WRITE(6,8)'LAT: ',OBSLOC(1),OBSLOC(2),OBSLOC(3)
      WRITE(6,8)'LONG: ',OBSLOC(4),OBSLOC(5),OBSLOC(6)
      WRITE(6,9)'HEIGHT: ',OBSLOC(7),'  HEADING: ',OBSLOC(8)
8     FORMAT(A8,I4,I3,I3)
9     FORMAT(A9,I7,A12,I3)
      WRITE(6,*)
      IF (FLAG .EQ. 1) THEN
        WRITE(6,*) 'ARE THERE ANY CORRECTIONS'
        WRITE(6,*)
      ENDIF
      WRITE(6,*)'SELECT ONE OF THE FOLLOWING'
        WRITE(6,*)'     (1) Latitude'
        WRITE(6,*)'     (2) Longitude'
        WRITE(6,*)'     (3) Height'
```

```
        WRITE(6,*)'     (4) Heading'
        WRITE(6,*)'     (5) All'
        WRITE(6,*)'     (6) None'
C
        READ(5,100) ANS
100     FORMAT(A1)
        IF (ANS .EQ. '6') GO TO 160
C
        IF (ANS .NE. '1' .AND. ANS .NE. '5') GO TO 60
          WRITE(6,*)'INPUT OBSERVER LATITUDE   -DEGREES: '
          READ(5,*)LATD
          WRITE(6,*)'                          -MINUTES: '
          READ(5,*)LATM
          WRITE(6,*)'                          -SECONDS: '
          READ(5,*)LATS
C
60      IF (ANS .NE. '2'.AND. ANS .NE. '5') GO TO 70
          WRITE(6,*)'INPUT OBSERVER LONGITUDE  -DEGREES: '
          READ(5,*)LOND
          WRITE(6,*)'                          -MINUTES: '
          READ(5,*)LONM
          WRITE(6,*)'                          -SECONDS: '
          READ(5,*)LONS
C
70      IF (ANS .NE. '3'.AND. ANS .NE. '5') GO TO 80
          WRITE(6,*)'INPUT OBSERVER HEIGHT     -METERS: '
          READ(5,*)HEIGHT
C
80      IF (ANS .NE. '4' .AND. ANS .NE. '5') GO TO 105
          WRITE(6,*)'INPUT THE COURSE           DEGREES: '
          READ(5,*)CTS
105     IF (ANS .LT. '1' .OR. ANS .GT. '6') GO TO 1
C
C load the values into the obs_loc array
C
150   OBSLOC(1)=LATD
      OBSLOC(2)=LATM
      OBSLOC(3)=LATS
      OBSLOC(4)=LOND
      OBSLOC(5)=LONM
      OBSLOC(6)=LONS
      OBSLOC(7)=HEIGHT
      OBSLOC(8)=CTS
      FLAG=1
      GO TO 1
C
C convert to XYZ coordinates
C
160   CALL DMS2XYZ(LATD,LATM,LATS,LOND,LONM,LONS,HEIGHT,X1,Y1,Z1)
C
C calc new position based on course
C
```

```
      IF (LATD .GT. 0) THEN
        LATS=5*COS(2*3.14159*CTS/360)+LATS
      ELSE
        LATS=LATS-5*SIN(2*3.14159*CTS/360)
      END IF
C
      IF (LOND .GT. 0) THEN
        LONS=5*SIN(2*3.14159*CTS/360)+LONS
      ELSE
        LONS=LONS-5*SIN(2*3.14159*CTS/360)
      END IF
C
C convert new position to XYZ coordinates
C
      CALL DMS2XYZ(LATD,LATM,LATS,LOND,LONM,LONS,HEIGHT,X2,Y2,Z2)
      RETURN
      END
```

```
C ************************************************************
      SUBROUTINE M_ORIEN(M,X1,Y1,Z1,X2,Y2,Z2)
C
C     THIS ROUTINE DETERMINES THE ANGLES OF ROTATION
C     OF THE IMAGE PLANE WITH RESPECT TO THE WORLD COORDINATE AXIES.
C     AND CALCULATES THE NEW M MATRIX FOR THE NEW VIEWER LOCATION
C
C     INPUTS  = X1,Y1,Z1          observer location
C               X2,Y2,Z2          observer direction point
C
C     OUTPUTS = M()               3D - 2D xform matrix
C
C ************************************************************
      IMPLICIT INTEGER (A-Z)
C
      REAL*8 MAGN_X,MAGN_Y,MAGN_Z
      REAL*8 X_VECX,X_VECY,X_VECZ,Y_VECX,Y_VECY,Y_VECZ
      REAL*8 Z_VECX,Z_VECY,Z_VECZ,X1,Y1,Z1,X2,Y2,Z2
      REAL*8 M(3,3)
C
C get the two OBS LOC points
C and set up vectors
C
      Y_VECX=X1
      Y_VECY=Y1
      Y_VECZ=Z1
C
C select another point along the track
C
      Z_VECX=X1-X2
      Z_VECY=Y1-Y2
      Z_VECZ=Z1-Z2
C
C use the cross product of Y CROSS Z to obtain the X vector
C
      X_VECX=((Y_VECY*Z_VECZ)-(Y_VECZ*Z_VECY))
      X_VECY=((Y_VECZ*Z_VECX)-(Y_VECX*Z_VECZ))
      X_VECZ=((Y_VECX*Z_VECY)-(Y_VECY*Z_VECX))
      MAGN_Z=SQRT((Z_VECX**2)+(Z_VECY**2)+(Z_VECZ**2))
      MAGN_X=SQRT((X_VECX**2)+(X_VECY**2)+(X_VECZ**2))
      MAGN_Y=SQRT((Y_VECX**2)+(Y_VECY**2)+(Y_VECZ**2))
      M(1,1)=X_VECX/MAGN_X
      M(1,2)=X_VECY/MAGN_X
      M(1,3)=X_VECZ/MAGN_X
      M(2,1)=Y_VECX/MAGN_Y
      M(2,2)=Y_VECY/MAGN_Y
      M(2,3)=Y_VECZ/MAGN_Y
      M(3,1)=Z_VECX/MAGN_Z
      M(3,2)=Z_VECY/MAGN_Z
      M(3,3)=Z_VECZ/MAGN_Z
      RETURN
      END
```

```
C
C ****************************************************************
C
      SUBROUTINE NEW_IJ(X1,Y1,Z1,X2,Y2,Z2,ITOT,
     .        XYZ,FOCUS,M,IMAX,IMAY,HIDDEN,DEPTH)
C
C     THIS SUBROUTINE COMPUTES THE NEW IA AND JA SCREEN
C     COORDINATES FROM THE GIVEN OBSERVER LOCATION.
C
C     INPUTS  = X1,Y1,Z1      observer location
C               X2,Y2,Z2      observer direction point
C               ITOT          # elev points
C               M()           3d - 2d xform matrix
C               FOCUS         focal length
C               XYZ()         dms coordinates of elev data
C
C     OUTPUTS = IMAX()        x image coordinate
C               IMAY()        y image coordinate
C               HIDDEN()      flag for points hidden behind FOV
C               DEPTH()       depth of each point
C
C ****************************************************************
      IMPLICIT INTEGER (A-Z)
C
      COMMON /ELEV/ ROW_ELEV,COL_ELEV,TOT_ELEV
C
      INTEGER ITOT,IR,HIDDEN(TOT_ELEV)
C
      REAL*8 X1,Y1,Z1,FOCUS,M(3,3)
      REAL*8 X,Y,Z,XIMA,YIMA,DENOM,DEPTH(TOT_ELEV)
      REAL*8 XYZ(TOT_ELEV,3),IMAX(TOT_ELEV),IMAY(TOT_ELEV)
      REAL*8 X2,Y2,Z2,OBS_VECX,OBS_VECY,OBS_VECZ
      REAL*8 OBJ_VECX,OBJ_VECY,OBJ_VECZ,MAG_OBS,MAG_OBJ
      REAL*8 COS_THETA,VALUE
C
      VALUE=90.0
      VALUE=COS(VALUE/57.29577951)
C
      DO 20 IR=1,ITOT
       X=XYZ(IR,1)
       Y=XYZ(IR,2)
       Z=XYZ(IR,3)
C
C calc the obs_vec
C
       OBS_VECX=X2-X1
       OBS_VECY=Y2-Y1
       OBS_VECZ=Z2-Z1
       MAG_OBS=SQRT(OBS_VECX**2+OBS_VECY**2+OBS_VECZ**2)
C
C calc the obj_vec
C
```

```
      OBJ_VECX=X-X2
      OBJ_VECY=Y-Y2
      OBJ_VECZ=Z-Z2
      MAG_OBJ=SQRT(OBJ_VECX**2+OBJ_VECY**2+OBJ_VECZ**2)
C
C calc cos theta, where theta is angle between line of sight
C and object point.
C
      COS_THETA=OBS_VECX*OBJ_VECX+OBS_VECY*OBJ_VECY+OBS_VECZ*OBJ_VECZ
      COS_THETA=COS_THETA/(MAG_OBS*MAG_OBJ)
C
C now check if COS(theta) > 0 ==> angle < 90 degrees
C
      IF (COS_THETA .GT. VALUE) THEN
        DENOM=M(3,1)*(X-X1)+M(3,2)*(Y-Y1)+M(3,3)*(Z-Z1)
        XIMA=-FOCUS*(M(1,1)*(X-X1)+M(1,2)*(Y-Y1)+M(1,3)*(Z-Z1))/DENOM
        YIMA=-FOCUS*(M(2,1)*(X-X1)+M(2,2)*(Y-Y1)+M(2,3)*(Z-Z1))/DENOM
        XIMA=XIMA*1000000.0
        YIMA=YIMA*1000000.0
C
C save xima, yima in IMAX(), IMAY() array temporarily
C
        IMAX(IR)=XIMA
        IMAY(IR)=YIMA
        DEPTH(IR)=SQRT(((X1-X)**2)+((Y1-Y)**2)+((Z1-Z)**2))
        HIDDEN(IR)=0
      ELSE
        HIDDEN(IR)=1
      END IF
20    CONTINUE
      RETURN
      END
```

```
C
C ***************************************************************
C
      SUBROUTINE XY2IJ(IA,JA,
     .      IMAX,IMAY,XFRAME,YFRAME,ITOT,HIDDEN)
C
C     THIS SUBROUTINE TAKES THE  IMAGE POINTS XIMA,YIMA AND
C     CONVERTS THEM TO I,J SCREEN POINTS
C     THIS IS THE AFFIN XFORM -- SEE REF 3
C
C     INPUTS  = IMAX()      x image coordinate
C               IMAY()      y image coordinate
C               XFRAME      image coordinate x axis size
C               YFRAME      image coordinate y axis size
C               ITOT        number of elev points
C               HIDDEN()    hidden point flag
C
C     OUTPUTS = IA()        screen x coordinate
C               JA()        screen y coordinate
C
C ***************************************************************
      IMPLICIT INTEGER (A-Z)
C
      COMMON /ELEV/ ROW_ELEV,COL_ELEV,TOT_ELEV
C
      INTEGER IA(TOT_ELEV),JA(TOT_ELEV),HIDDEN(TOT_ELEV),ITOT
C
      REAL*8 XIMA,YIMA,A1,A2,B1,B2,C1,C2,DENOM
      REAL*8 IMAX(TOT_ELEV),IMAY(TOT_ELEV),XFRAME,YFRAME
C
      DATA I_MAX,J_MAX/512.0,512.0/
C
C calc the affin parameters
C
      A1=XFRAME/(I_MAX*1.0)
      A2=0.0
      B1=0.0
      B2=YFRAME/(J_MAX*1.0)
      C1=-XFRAME/2.0
      C2=-YFRAME/2.0
C
        DO 25 IR=1,ITOT
          IF (HIDDEN(IR) .EQ. 1) GO TO 25
          XIMA=IMAX(IR)
          YIMA=IMAY(IR)
          IA(IR)=(XIMA-C1)/A1
          JA(IR)=(YIMA-C2)/B2
25        CONTINUE
       RETURN
       END
```

```
C
C ***************************************************************
C
      SUBROUTINE HIDDEN_SURF(IA,JA,IENDM,IENDN,DEPTH,
     .      PL_AR,VD_ID,HIDDEN,DEPTH_KEY,WD_ID)
C
C     THIS ROUTINE WILL CHECK EACH PANEL AGAINST ALL
C     OTHERS FOR OVERLAP.
C     INPUTS ==> IA( )
C                JA( )
C                IENDN
C                IENDM
C                HIDDEN()   flag for points hidden behind FOV
C
C      OUTPUTS = NONE
C ***************************************************************
C
      IMPLICIT INTEGER (A-Z)
C
      COMMON /ELEV/ ROW_ELEV,COL_ELEV,TOT_ELEV,NPLANES
C
      INTEGER IGREY,NODE_A,NODE_B,NODE_C
      INTEGER IENDN,IENDM,IR,I,J
      INTEGER IPLANES,HIDDEN(TOT_ELEV),PL_AR(NPLANES,5)
      INTEGER IA(TOT_ELEV),JA(TOT_ELEV),DEPTH_KEY(NPLANES)
C
      INTEGER X1,Y1,X2,Y2,X3,Y3
      REAL*8 DEPTH(TOT_ELEV)
C
      LOGICAL SORTED,PLUS,MINUS
C
C calculate the number of planes
C
      IPLANES=((IENDM-1)*2)*(IENDN-1)
      COUNT=0
C
C read in the data for each plane
C
      DO 100 IR=1,IPLANES
        PTA=PL_AR(IR,1)
        PTB=PL_AR(IR,2)
        PTC=PL_AR(IR,3)
        IF (HIDDEN(PTA) .EQ. 1 .OR. HIDDEN(PTB) .EQ. 1 .OR. HIDDEN(PTC)
     .      .EQ. 1) THEN
          GO TO 100
        ELSE
          COUNT=COUNT+1
          DEPTH_KEY(COUNT)=IR
        ENDIF
100   CONTINUE
C
C sort depth values
```

```
C
      IPLANES=COUNT
      DO 105 L=1,IPLANES
        IR=DEPTH_KEY(L)
        PL_AR(IR,5)=MAX(DEPTH(PL_AR(IR,1)),
     .    DEPTH(PL_AR(IR,2)),DEPTH(PL_AR(IR,3)))
105   CONTINUE
C
       CALL QUICKSORT(PL_AR,IPLANES,DEPTH_KEY)
C
       CALL UISDC$ERASE (WD_ID)
C
C now begin selection
C
109   DO 110 K=IPLANES,1,-1
        IR=DEPTH_KEY(K)
C
C draw to virtual display
C
        NODE_A=PL_AR(IR,1)
        NODE_B=PL_AR(IR,2)
        NODE_C=PL_AR(IR,3)
        IGREY=PL_AR(IR,4)
C
C limit the gray scale to 250 shades of gray
C vice 256 shades
C
        IF (IGREY .EQ. 0) IGREY=1
        IF (IGREY .GT. 250) IGREY=250
        X1=IA(NODE_A)
        Y1=JA(NODE_A)
        X2=IA(NODE_B)
        Y2=JA(NODE_B)
        X3=IA(NODE_C)
        Y3=JA(NODE_C)
        CALL UIS$SET_WRITING_INDEX(VD_ID,1,1,IGREY)
        CALL UISDC$PLOT(WD_ID,1,X1,Y1,X3,Y3,X2,Y2)
C
110     CONTINUE
        RETURN
        END
```

```
C
C ***********************************************
C
        SUBROUTINE CLRSCRN
C
C    This routine clears the screen
C
C    Inputs = NONE
C
C    Outputs = NONE
C
C ***********************************************
        IMPLICIT INTEGER (A-Z)
        CALL SMG$CREATE_PASTEBOARD(PB_ID)
        CALL SMG$ERASE_PASTEBOARD(PB_ID)
        RETURN
        END




C
C *****************************************************
C
        SUBROUTINE CONV2SEC(DEG,MIN,SEC)
C
C        INPUTS = DEG
C                 MIN
C                 SEC
C
C        OUTPUTS= SEC
C
C *****************************************************
        IMPLICIT INTEGER (A-Z)
        IF (DEG .GE. 0) THEN
          SEC=SEC+MIN*60+DEG*3600
        ELSE
          SEC=DEG*3600-MIN*60-SEC
        ENDIF
        RETURN
        END
```

```
C
C ***************************************************
      SUBROUTINE QUICKSORT(ARRAY,COUNT,KEY)
C
C     INPUT = ARRAY()  array to sort, sort on 5th field
C             KEY()    key index to main array
C             COUNT    number of elements to sort
C
C     OUTPUT= KEY()    new key index to main array
C
C ***************************************************
C
      IMPLICIT INTEGER (A-Z)
C
      COMMON/ELEV/NN,MM,TOT,NPLANES
      INTEGER ST(100,2)
      INTEGER COUNT,KEY(NPLANES)
      INTEGER ARRAY(NPLANES,5)
C
      SP=1
      ST(1,1)=1
      ST(1,2)=COUNT
      DO WHILE (SP .NE. 0)
        L=ST(SP,1)
        R=ST(SP,2)
        SP=SP-1
        DO WHILE (R .GT. L)
          LI=L
          RI=R
          INDEX=INT((L+R)/2)
          SA=ARRAY(KEY(INDEX),5)
          DO WHILE (LI .LE. RI)
            DO WHILE (ARRAY(KEY(LI),5) .LT. SA)
              LI=LI+1
            END DO
            DO WHILE (ARRAY(KEY(RI),5) .GT. SA)
              RI=RI-1
            END DO
            IF (LI .LE. RI) THEN
              TEMP=KEY(LI)
              KEY(LI)=KEY(RI)
              KEY(RI)=TEMP
              LI=LI+1
              RI=RI-1
            ENDIF
          END DO
          IF ((R-LI) .GT. (RI-L)) THEN
            IF (L .LT. RI) THEN
              SP=SP+1
              ST(SP,1)=L
              ST(SP,2)=RI
            ENDIF
```

```
        L=LI
      ELSE
        IF (LI .LT. R) THEN
          SP=SP+1
          ST(SP,1)=LI
          ST(SP,2)=R
        ENDIF
        R=RI
      ENDIF
    END DO
  END DO
  RETURN
  END
```

```
C************************************************
C
      SUBROUTINE IMAGE_SAVE(WD_ID,FILE_NAME,FLAG)
C
C************************************************
C
      IMPLICIT INTEGER(A-Z)
C
      CHARACTER FILE_NAME*(*),NAME1*23,NAME2*23
C
      INTEGER WD_ID,FLAG
C
      DATA RETLEN/0/
      DATA RWIDTH/0/
      DATA RHEIGHT/0/
      DATA BPPIXEL/0/

C
C determine the size of the image buffer
C
      CALL UISDC$READ_IMAGE(WD_ID,0,0,512,512,RWIDTH,RHEIGHT,BPPIXEL,
     .     ENCODED1,0)
C
      RETLEN=RWIDTH*RHEIGHT
C
C open the external files
C
      IF (FLAG .EQ. 1) THEN
        C1=INDEX(FILE_NAME,'.')
        IF (C1 .EQ. 0) THEN
          C2=INDEX(FILE_NAME,' ')
          NAME1=FILE_NAME(1:C2-1)//'.SIZ'
          NAME2=FILE_NAME(1:C2-1)//'.IMG'
        ELSE
          NAME1=FILE_NAME(1:C1-1) // '.SIZ'
          NAME2=FILE_NAME
        ENDIF
        WRITE(6,*)NAME1,NAME2
        OPEN(UNIT=11,FILE=NAME1,STATUS='NEW',
     .       ACCESS='SEQUENTIAL',FORM='UNFORMATTED')
C
        OPEN(UNIT=10,FILE=NAME2,STATUS='NEW',
     .       FORM='UNFORMATTED')
        WRITE(11)RWIDTH,RHEIGHT,BPPIXEL,RETLEN
C
C allocate virtual memory for the buffer
C
        STATUS=LIB$GET_VM(RETLEN,ENCODED)
        IF (.NOT. STATUS) CALL LIB$STOP(%VAL(STATUS))
      ENDIF
C
C extract and store private data in buffer
```

```
C
      CALL UISDC$READ_IMAGE(WD_ID,0,0,512,512,RWIDTH,RHEIGHT,BPPIXEL,
     .    %VAL(ENCODED),RETLEN)
C
C call subroutine to write the contents of the buffer
C
      CALL BUFFERWRITE(%VAL(ENCODED),RETLEN,10)
C
      RETURN
      END


C
C ********************************************
C
      SUBROUTINE BUFFERWRITE(BUFFER,LENGTH,LUN)
C
C ********************************************
C
      IMPLICIT INTEGER(A-Z)
C
      BYTE BUFFER(LENGTH)
C
      WRITE (LUN)BUFFER
C
      RETURN
      END
```

LIST OF REFERENCES

1. Furness, Thomas A., "Fantastic Voyage", Popular Mechanics, v. 163, #12, Dec. 1986.

2. Coleman, Leland G., Three-Dimensional Image Generation From an Aerial Photograph, Thesis, Naval Postgraduate School, Monterey, California, Sep. 1987.

3. Naval Postgraduate School Report NPS52-87-034, An Inexpensive Real-Time Interactive Three-Dimensional Flight Simulation System, by M. J. Zyda, and others, Jul. 1987.

4. Moffit, Francis H., Mikhail, Edward M., Photogrammetry, 3rd Edition, Harper & Row, 1980.

5. EEZ-SCAN 84 Scientific Staff, Atlas of the Exclusive Economic Zone, Western Conterminous United States, U.S. Geological Survey Miscellaneous Investigations Series I-1792, 1986.

6. Tyce, Robert C., "Deep Seafloor Mapping Systems-A Review", MTS Journal, v. 20, #4, Dec. 1986.

7. Chavez, S. Pat Jr., "Processing Techniques for Digital Sonar Images from GLORIA", Photogrammetric Engineering and Remote Sensing, v. 52, #8, Aug. 1986.

8. Hearn, Donald and Baker, M. Pauline, Computer Graphics, Prentice-Hall, 1986.

9. Rodgers, David F., Procedural Elements for Computer Graphics, McGraw-Hill Book Co., 1985.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, VA 22304-6145	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, CA 93943-5002	2
3.	Professor Chin-Hwa Lee Code 62 Le Electrical and Computer Engineering Department Naval Postgraduate School Monterey, CA 93943	15
4.	Chairman, Code 62 Electrical and Computer Engineering Department Naval Postgraduate School Monterey, CA 93943	1
5.	Professor C. W. Therrien Code 62 Ti Electrical and Computer Engineering Department Naval Postgraduate School Monterey, CA 93943	1
6.	RADM. G. H. Curtis Naval Sea Systems Command (PMS 350) National Center #3 2531 Jefferson Davis Highway Arlington, VA 22202	1
7.	Dr. Arthur Bisson Assistant Technical Director, Acoustics Strategic Submarine Division (CNO/OP-02) Crystal Mall #3 1931 Jefferson Davis Highway Arlington, VA 22202	1
8.	Lt. Virginia Oard Defense Mapping Agency Headquarters U.S. Naval Observatory BLDG. 56 Washington, D.C. 20305-3000	1

9. Director, Defense Mapping Agency 1
Hydrographic/Topographic Center
ATTN: James C. Hammack (code REA)
6500 Brookes Lane
Washington, D.C. 20315

10. Commanding Officer 1
Naval Oceanographic Office
Bay St.
Louis, MS 39522-5001

11. Lt. Robert J. Myers 2
c/o Robert Myers
R.D. #4 Box 328
Washington, PA 15301

Thesis
M99481 Myers
c.1 Three-dimensional perspective image generation from sonar bathymetry and imagery data.